WHOI-99-13

Woods Hole Oceanographic Institution

Visual: A Visualization System for Accessing and Analyzing Multi-Sensor Data

by

Steve Lerner

August 1999

Technical Report

Funding was provided by the National Science Foundation under Grant No. OCE-9627160.

WHOI-99-13

Visual: A Visualization System for Accessing and Analyzing Multi-Sensor Data

by

Steve Lerner

Woods Hole Oceanographic Institution
Woods Hole, Massachusetts 02543

August 1999

Technical Report

Funding was provided by the National Science Foundation under contract number OCE-9627160.

Approved for Distribution:

Timothy K. Stanton

Timothy K. Stanton, Chair

Department of Applied Ocean Physics and Engineering

Visual: A Visualization System for Accessing and Analyzing Multi-Sensor Data

1 Background

1.1 Overview

Visual is a visualization system used primarily to access and analyze high-volume multi-sensor data collected from remotely operated underwater vehicles. Since 1991, scientists have used *Visual* for scientific visualization and analysis of underwater surveys ranging from real-time survey monitoring[1], to geological mapping and interpretation of hydrothermal vent sites[2,3,4], to a forensic study of a shipwreck[5]. *Visual* was developed by Steve Lerner at the Deep Submergence Laboratory of Woods Hole Oceanographic Institution with contributions by Marty Marra, Jonathan Howland, and Bill Lange. The system was originally developed in 1991 to provide real-time visualization of side-scan imagery overlaid onto bathymetry for use with the DSL120 sonar system[6]. Since then, *Visual* has evolved into a general visualization program that goes beyond 3-dimensional fly-throughs of sonar data. *Visual*'s capabilities include the following:

1) 2–dimensional and 3-dimensional viewing of multi-sensor data (sonar data, images, etc.).
2) A spatial interface to multi-sensor data (images, video, navigation, core samples, url's, etc.).
3) The capability to easily access and review large datasets interactively (100,000+ objects).
4) A real-time interface for at-sea operations, telepresence, or playback.
5) The ability to handle analog imagery such as film via a laser disk and videotape interface.
6) Support for a 6-degree of freedom input device (Spaceball) and stereographic display.
7) External application interface to easily extend *Visual*'s capabilities.

Typical applications for *Visual* include sonar visualization, real-time monitoring, and multi-sensor data access and analysis.

1.2 Sonar Visualization

For sonar visualization, *Visual* provides the capability of interactive 3-dimensional fly-throughs of side-scan imagery overlaid onto bathymetry. Having the ability to interactively pan, zoom, and fly-around allows the terrain to be viewed from any perspective. To view the terrain in color, the black-and-white side-scan imagery can be combined with a color scale for bathymetry. *Visual* does this by modulating the intensity of the terrain with the amplitude of the backscatter, while the hue is modulated by the bathymetry. For large datasets, *Visual* can dynamically change the resolution to maintain interactive motion, and when stopped, increase the resolution enabling a particular region to be studied in full detail. Finally, the depth information as well as measuring objects can be accomplished simply by moving the cursor over the terrain. The sonar data itself may be used as a background map for real-time monitoring and multi-sensor data access. Examples of sonar data visualizations are shown for the Juan de Fuca Ridge (Crest91), an Arctic Under Ice Survey (Simi93), and Tag Hydrothermal Mound (Tag94) in Figures 1-1, 1-2, 1-3 respectively.

1.3 Real-time Monitoring

Telepresence operations and Situational Awareness Displays are two examples of *Visual* real-time monitoring applications. *Visual* provides a real-time interface for vehicle telemetry (navigation and attitude information) and real-time data (e.g., temperature). Objects such as CAD vehicle models and navigation tracks can be overlaid onto a terrain map. Multiple viewpoints such as a 3D-perspective world-view, vehicle-view, or plan-view are available and are user selectable. The background map may be sonar data, image data such as mosaics, etc. The real-time data can be logged and played back at a later time. Examples of real-time monitoring applications for the Jason Project in Guaymas Basin (Guaymas93), WHOI Dock Testing (Dock94), and the Situational Awareness Display for the Derbyshire Survey (Derby97) are shown in Figures 1-4, 1-5, 1-6.

1.4 Multi-Sensor Data Access and Analysis

Visual excels at providing access to high-volume multi-sensor data, either spatially or temporally. The spatial interface is the most common method and allows one to directly select objects overlaid onto a background map. Combined with the interactive panning and zooming capabilities along with customizable background maps, it is very easy to navigate to a particular region of interest. One can then set a search radius and select all objects (images, navigation tracks, etc.) within that region and display the resulting imagery, data, etc. Note: As the cursor moves over objects, information about those objects is displayed in the information window. The objects themselves may be represented as points, lines, polygons, sensor footprints, image icons, or text. The background maps may be 2 or 3 dimensional sonar data, digitized maps, or images including mosaics.

With the laser disk interface, one can play back digital images temporally (i.e., in the order that they were collected). Depending on the rate that the images were recorded on the disk, the system is capable of playing back the images several times faster than real-time (up to 60 frames/second). During playback, the images are displayed on an external monitor and an icon is simultaneously displayed on the main *Visual* window highlighting the corresponding location of the image. With this setup, it is possible to preview up to 45,000 images fairly quickly. Note: We are currently looking at implementing a software emulator for the laser disk system that will provide the same functionality with on-line digital images.

Examples of high-volume multi-sensor datasets include the Derbyshire survey (Derby97) and the Trans-Atlantic Geotraverse survey (Tag94). Refer to Figures 1-7, 1-8a, 1-8b. Finally, *Visual* can also be used to aid in navigation verification[7] (refer to Figure1-9).

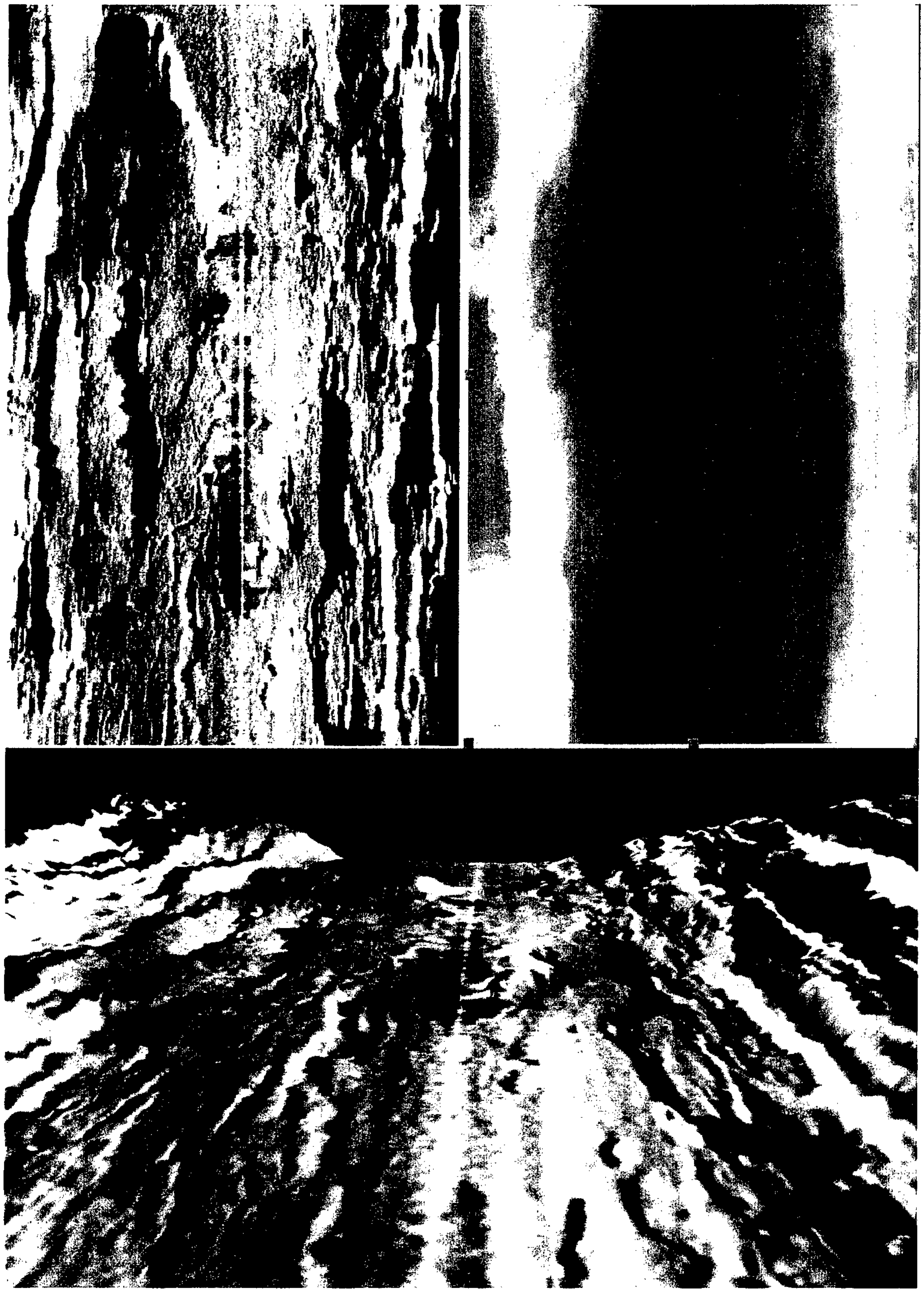


Figure 1-1: Sonar Visualization - Juan de Fuca Ridge (Crest91)

A) DSL120 sidescan imagery of 1km swath of axial valley (top-left image)
B) Co-registered bathymetry, ranging in depth from 2082-2238 meters (top-right image)
C) 3D View of sidescan imagery textured-mapped over bathymetry (bottom image)

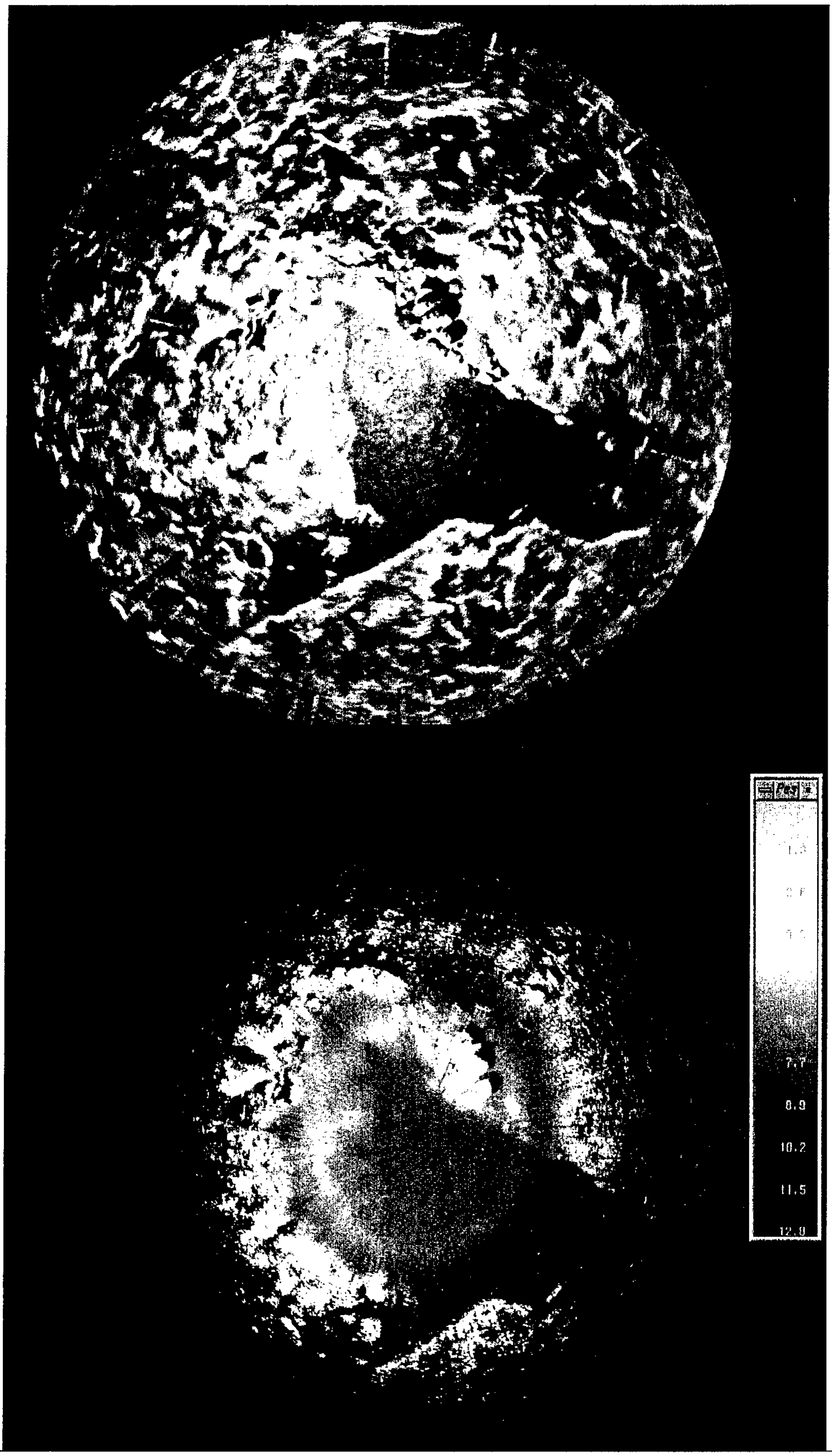


Figure 1-2: Sonar Visualization - Arctic Under Ice Survey (Simi93)

Researchers lowered a DSL200 kHz sidescan transducer through a hole in an Arctic ice floe to map the underside structure of the ice. Mounted on a rotator device, the top image shows the 200m radius sidescan imagery with the co-registered bathymetry shown in the bottom image.

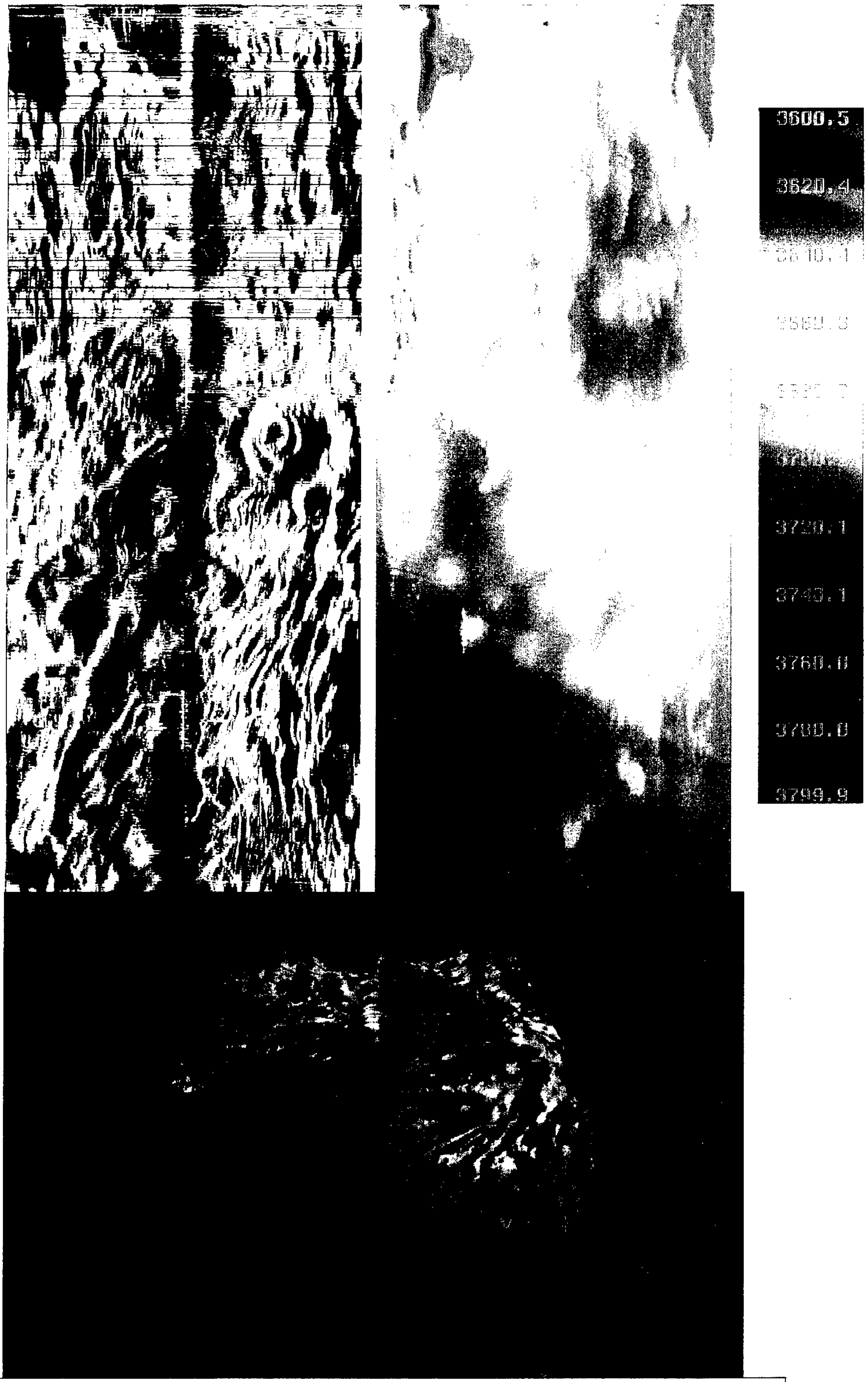
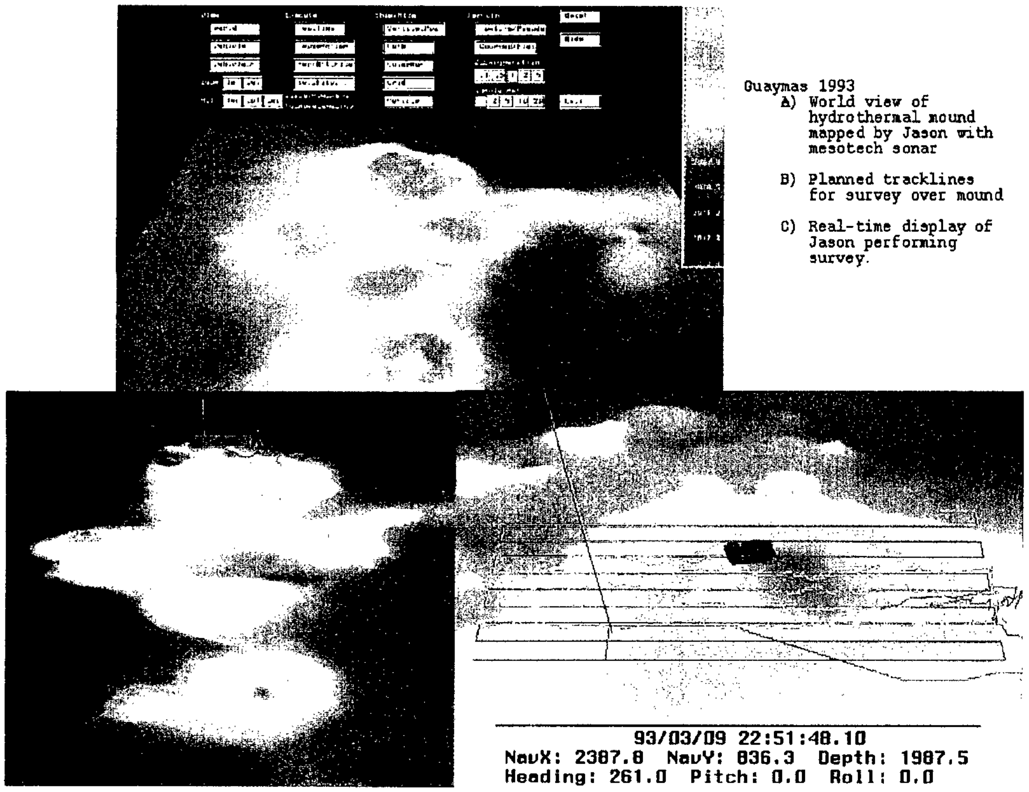


Figure 1-3: Sonar Visualization -TAG Hydrothermal Mound (TAG94)

A) DSL120 sidescan imagery of 1km swath (top-left image)
B) Co-registered bathymetry with depth scale (top-right images)
C) 3D Color View of sidescan imagery textured-mapped over bathymetry (bottom image)



```
Guaymas 1993
    A) World view of
       hydrothermal mound
       mapped by Jason with
       mesotech sonar

    B) Planned tracklines
       for survey over mound

    C) Real-time display of
       Jason performing
       survey.
```

```
           93/03/09 22:51:48.10
NavX: 2387.8  NavY: 836.3  Depth: 1987.5
Heading: 261.0  Pitch: 0.0  Roll: 0.0
```

Figure 1-4: Telepresence Application - Jason Project Guaymas 1993 (Guaymas93)

A composite of three different views as seen on *Visual*. Real-time data including vehicle navigation, attitude, and temperature were broadcast over the Internet and displayed on *Visual*. In addition to the above displays, scientists could simultaneously view a sensor window displaying temperature along with a live video link showing Jason's video imagery. With the combination of these displays and two-way voice communication, scientists at Woods Hole Oceanographic Institution were able to conduct a remote temperature survey over a hydrothermal mound in Guaymas, Mexico. Note: The track-line separation is approximately 1m.

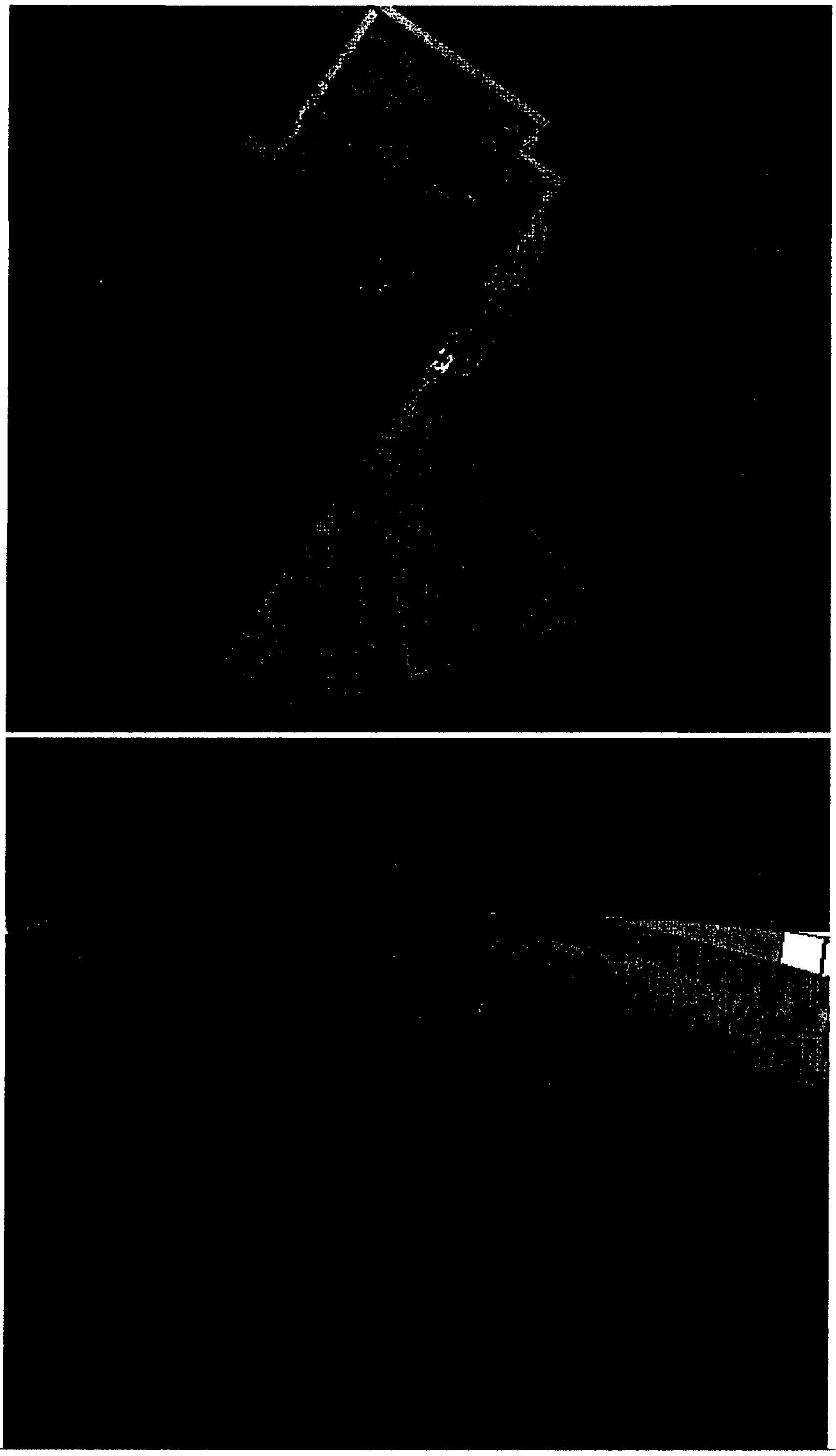
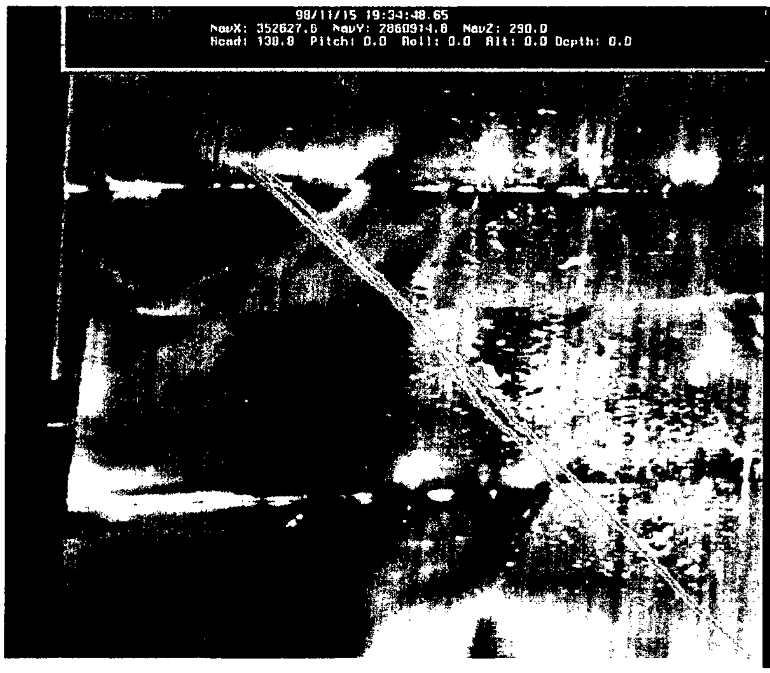


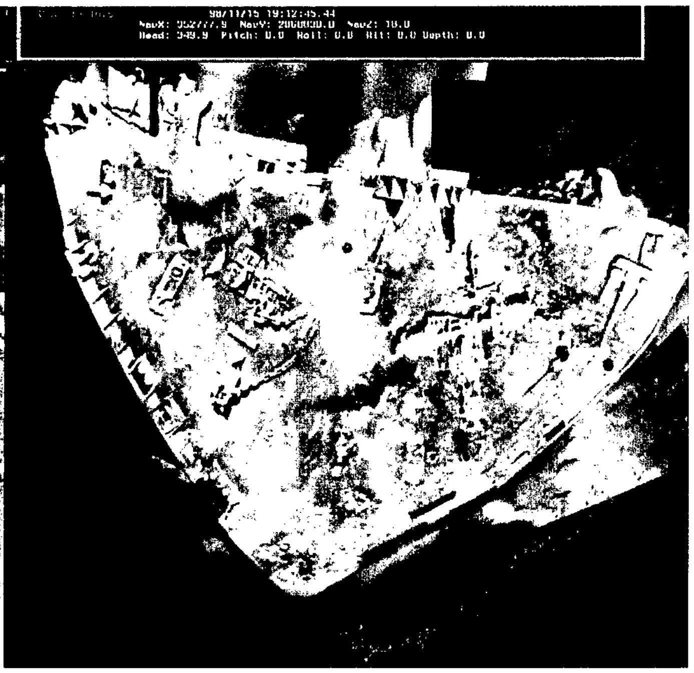
Figure 1-5: Real-Time Situational Awareness Display - WHOI Dock Trials (Dock94)

Example of visualizing multi-sensor data in a real-time environment for remote operations.

Dock trials using Jason equipped with a Seabat multi-beam sonar and a scanning imagenics sonar (red). Bathymetry data is combined with imagery from the DSL200kHz sidescan sonar collected with a rotator device mounted on the dock in 1993. The radius of the sonar swath is 100 meters. The WHOI dock, seawall, and the MBL dock are CAD models taken from drawings. Note the offsets between the real-time data and the CAD models near the left-hand side of the seawall. This is due to the CAD models assumption that the angles of the seawall were 90 degrees. An actual navigation track is also displayed.

This display was used as a remote operations display in the WHOI Blake building as dock trials were being conducted. Real-time navigation and attitude data were broadcast over the Internet.


98/11/15 19:34:48.65
NavX: 352627.6 NavY: 2860914.8 NavZ: 290.0
Head: 130.8 Pitch: 0.0 Roll: 0.0 Alt: 0.0 Depth: 0.0


98/11/15 19:12:45.44
NavX: 352777.9 NavY: 2860030.0 NavZ: 18.0
Head: 349.9 Pitch: 0.0 Roll: 0.0 Alt: 0.0 Depth: 0.0

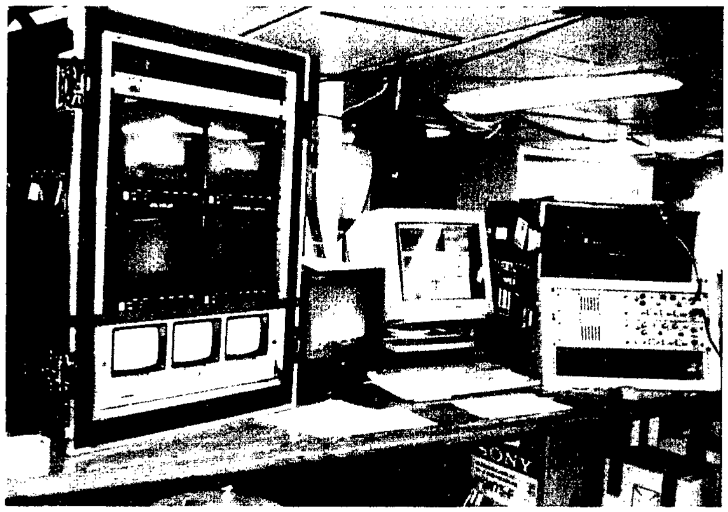
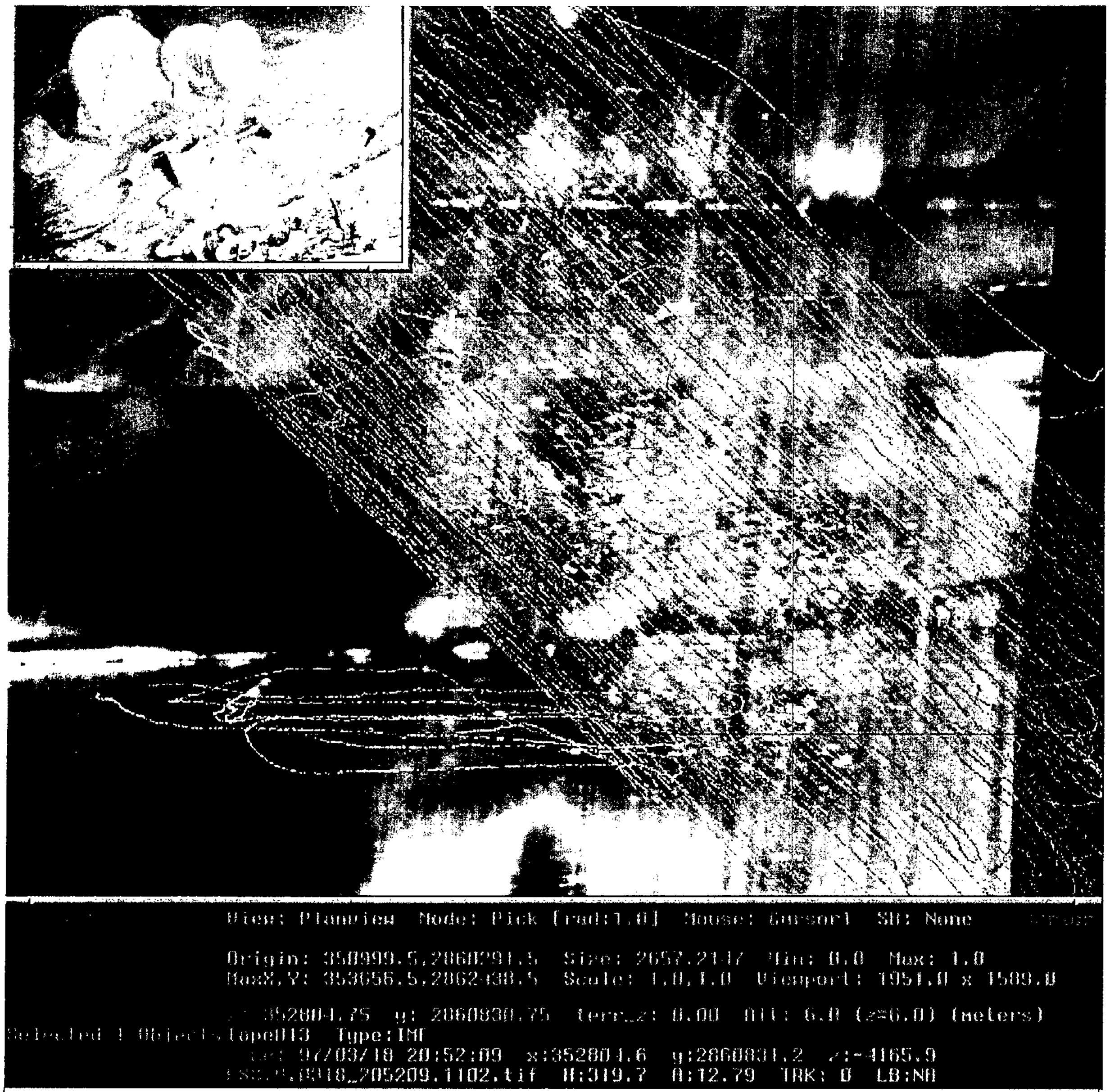


Figure 1-6: Real-Time Situational Awareness Display – Derbyshire 1997

A) Real-Time Observation Station for the Derbyshire 1997 survey consisting of multi-channel video display, digital still image monitor, and the *Visual* Situational Awareness Workstation (bottom image).

B) *Visual* Situational Awareness Display showing real-time ArgoII navigation tracks overlaid onto DSL120 side-scan imagery background. ArgoII real-time navigation and attitude information is also displayed on top-portion of screen. Real-time video and digital images were simultaneously displayed on a multi-channel video display located next to the *Visual* workstation (top-left image).

C) *Visual* Situational Awareness Display showing Jason performing close-in inspection of Derbyshire bow section. Jason real-time navigation and attitude information is displayed on top-portion of screen and real-time navigation tracks are overlaid onto bow mosaic. In addition, real-time video and digital images were simultaneously displayed on a multi-channel video display located next to the *Visual* workstation (top-right image).



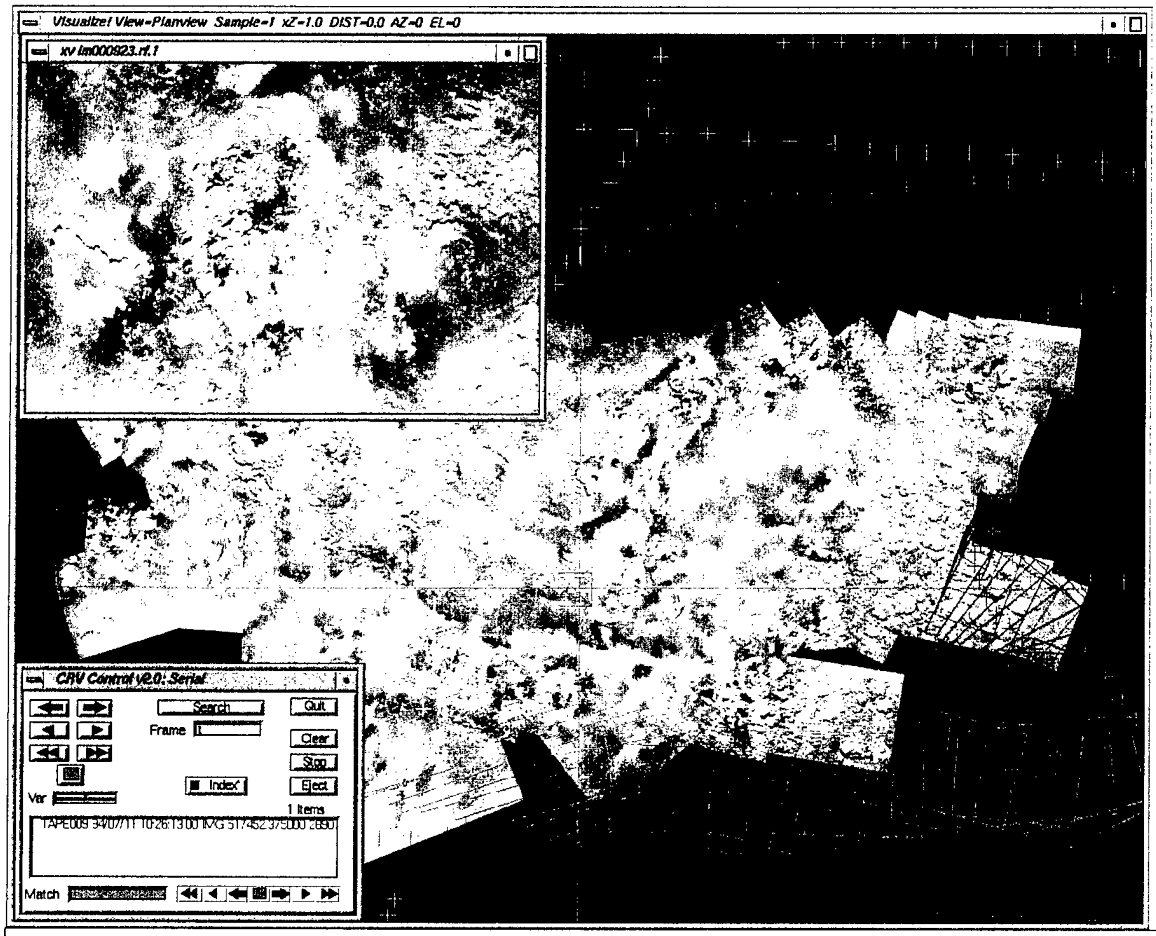
```
                View: Planview  Mode: Pick [rad:1.0]  Mouse: Cursor1   SB: None

                Origin: 350999.5,2860291.5  Size: 2657,2147  Min: 0.0  Max: 1.0
                MaxX,Y: 353656.5,2862438.5  Scale: 1.0,1.0  Viewport: 1951.0 x 1589.0

                x: 352804.75  y: 2860830.75  terr_z: 0.00  Alt: 6.0 (z=6.0) (meters)
Selected 1 Objects Lope013  Type:IMF
                   97/03/18 20:52:09  x:352804.6  y:2860831.2  z:-4165.9
                ESC.0318_205209.1102.tif  H:319.7  A:12.79  TRK: 0  LB:NA
```

Figure 1-7: Multi-Sensor Data Access and Analysis - Derbyshire 1997

Locations of ARGOII digital still camera images overlaid onto DSL120 Side-Scan Imagery. Over 135,000 image locations are shown. The crosshair cursor is set to a 1-meter selection radius. The information window below the main *Visual* window displays information about the *Visual* window, cursor location, and selected objects. Objects can be selected and the corresponding images of those selected objects can be displayed in the upper left-hand corner or on a laser disk video monitor. Having a direct spatial access is critical for a survey like this since an area of interest may have been mapped or imaged several times but the data may have been collected over several days or even weeks apart.



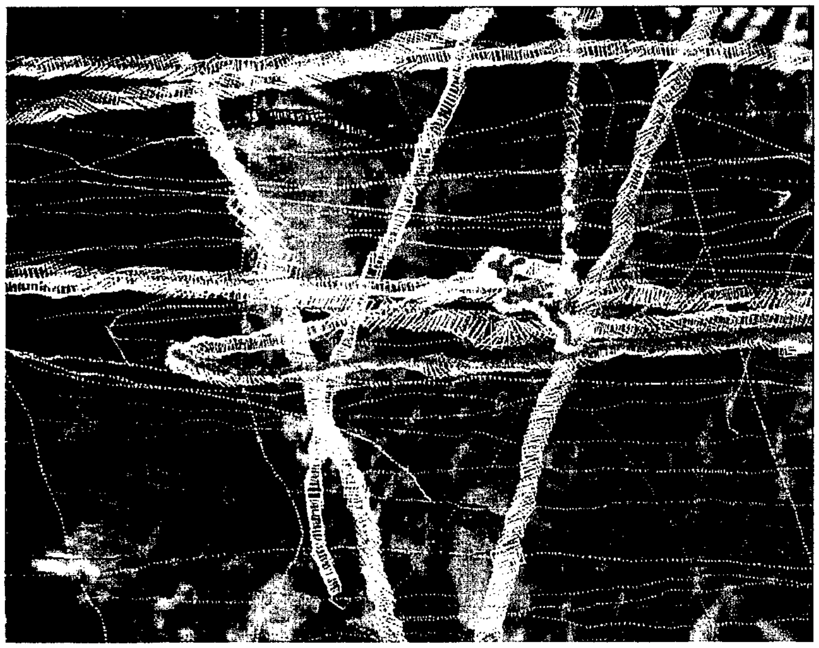
TAPE009 94/07/11 10:26:13.00 IMG 517452 375000 2890

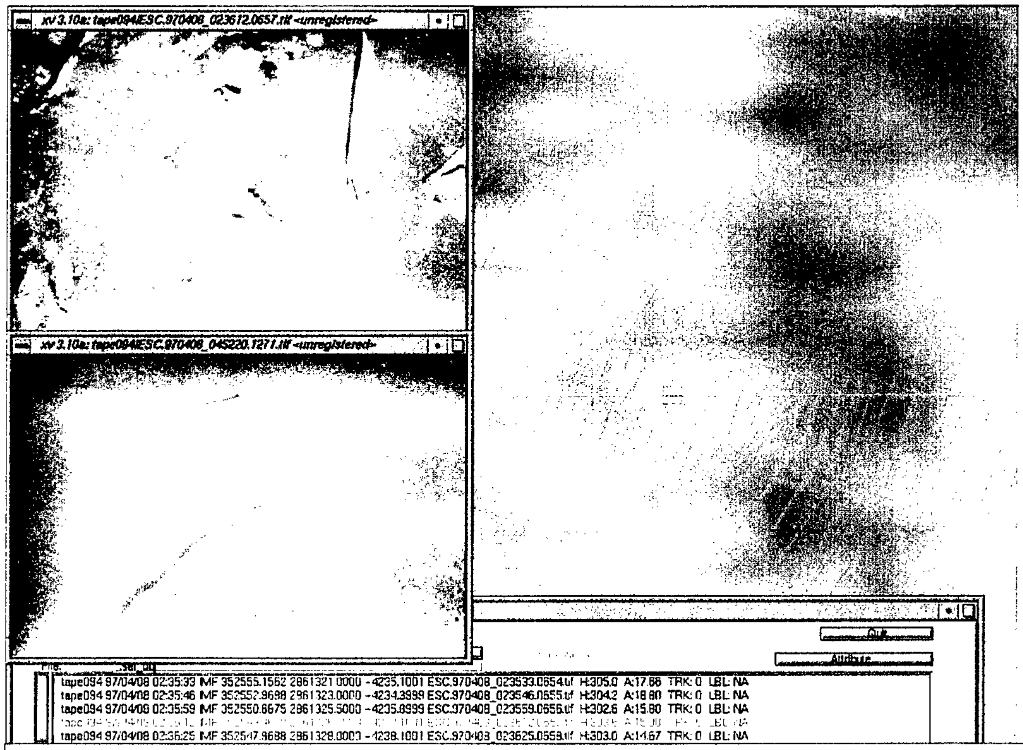
Figure 1-8a: Multi-sensor Data Access and Analysis - Tag94 with photo-mosaic background

In this example, footprints of overlapping images are displayed onto a photo mosaic that was previously created. Using this method, the 'big picture' representation is presented via the base mosaic, and at the same time one can access and analyze the original photographs. In mosaics, choices are made as to which image should be used and typically these images are warped in a way that makes the final mosaic aesthetically pleasing but hard to do quantitative measurements. By having access to the original images, one can go back at look at other images in the area as well the ability to measure and analyze features accurately. For each image selected, information about the image such as position, heading, altitude, and descriptions are displayed in the information window. In this example, note that the orientation of the image in the upper-left is different than the orientation in the mosaic. This is because the image shown in the upper left is displayed in the orientation that it was collected in; there is no automatic north rotation or scaling being performed on the image. Future versions of the image viewer will have the option to present the image in a north-up orientation.

Figure 1-8b: Multi-sensor Data Access and Analysis – Tag94

Locations of ARGOII high-resolution digital still camera images and 35mm photographic imagery overlaid onto a shaded-relief DSL120 side-scan bathymetric map. 35mm film image locations are shown as red asterisks. Colors of the track-lines are based on the data tapes that the images were recorded to in real-time. Approximately 1000 images were recorded per tape with over 30,000 images collected. The image footprint coverage is shown as rectangles and can be toggled on or off interactively. Both the digital and 35mm film images were recorded to a laser disk system and are directly accessible via *Visual*. The TAG hydrothermal mound is approximately 200m in diameter at a depth of 3650m.





xv 3.10a: tape094ESC.970408_023612.0657.tif <unregistered>
xv 3.10a: tape094ESC.970408_045220.1271.tif <unregistered>
Quit
Attribute

```
tape094 97/04/08 02:35:33 MF 352555.1562 2861321.0000 -4235.1001 ESC.970408_023533.0654.tif H:305.0 A:17.66 TRK: 0 LBL: NA
tape094 97/04/08 02:35:46 MF 352552.9698 2861323.0000 -4234.3999 ESC.970408_023546.0655.tif H:304.2 A:18.80 TRK: 0 LBL: NA
tape094 97/04/08 02:35:59 MF 352550.6675 2861325.5000 -4235.8999 ESC.970408_023559.0656.tif H:302.6 A:15.80 TRK: 0 LBL: NA
[illegible]
tape094 97/04/08 02:36:25 MF 352547.9688 2861328.0000 -4238.1001 ESC.970408_023625.0658.tif H:303.0 A:14.67 TRK: 0 LBL: NA
```

Figure 1-9: Using *Visual* for Navigation Verification

In this example, footprints of overlapping images are displayed on the right-hand side. Two images have been selected and are displayed on the left-hand side. You can see the matching features although they are in different orientations. The navigation information about each of the images is shown in the bottom window and can be compared to the visual offsets of target points seen within the images.

2 User Manual

2.1 System Requirements

Visual is written in C and runs on Silicon Graphics computers. Supported computer platforms include the Silicon Graphics Indigo2 and O2. For performance of texture-map applications it is recommended that a computer with hardware texture-map capabilities be used. Performance will be significantly increased with extra memory. A minimum of 128MB of memory is recommended and 256MB or more is preferred.

2.2 Running Visual

Once you are logged into a *Visual* account that has been set up during the installation process, there are several ways to run *Visual* depending on how the datasets are prepared. The three common methods for running *Visual* include:

1) A Netscape Interface – Datasets may be loaded and executed with Netscape
2) A command line script: visual.csh. Usage visual <dataset name><cr>
3) A dataset command file. Type: `cd <dataset_name><cr> <dataset_name.cmd>` <cr>

Each method has its own advantages and disadvantages, and it will be up to either the user or the *Visual* system administrator as to which method is appropriate. Note: How *Visual* begins executing is strictly an administrative choice as there is no difference once *Visual* is running. As *Visual* begins to execute, several messages may appear on the screen indicating the progress of loading the dataset if the verbose flag has been set. Shortly thereafter, a blue screen will appear with a Woods Hole Oceanographic Institution logo in the bottom left-hand corner of the display. Finally, the dataset will appear in the last saved configuration.

Refer to Figure 2-1, 2-2 for examples of the Netscape Interface to *Visual*. The Netscape Interface is the easiest to use since it is simply a point and click operation. The disadvantage to using the Netscape Interface is that it requires a machine with extra memory to run both *Visual* and Netscape and the datasets of interest must be pre-loaded into the Netscape dataset html page. There are tools to simplify building the dataset html page (refer to the Reference Guide).

When executing *Visual* via the visual.csh command, you can list the available datasets by typing visual.csh with no arguments. Below is a description of the C-shell program visual.csh along with its usage.

```
visual.csh is a C shell wrapper to the visual executable. The program provides an easy
interface to execute visual with a particular dataset. When new datasets have been
collected or wish to be added to the list, simply edit the visual.csh program and add it.
The usage of visual.csh follows.

Usage: visual.csh <name> [optional params]
       crest16   - Juan de Fuca Ridge Sidescan Imagery/Bathymetry
       dock94    - WHOI Dock94
       guaymas   - Jason Project in the Sea of Cortez
       monitor   - USS Monitor
       simi93    - Arctic - Sea Ice Mechanics Initiative
       srp93     - Mid-Atlantic Ridge
       tag94     - TAG Hydrothermal Site Sidescan Bathymetry
       tag94.ice - TAG Hydrothermal Site with ESC images
       tag94.crv - TAG Hydrothermal Site with ESC images and CRV
       tag94.shade.crv - Same as above but with shaded-relief background
       visual.doc - Visual Documentation via Netscape Browser

Example: visual.csh tag94
```

Finally, the last method for running *Visual* is to use a *Visual* dataset.cmd file. This provides the most flexibility, but requires some knowledge of navigating through a directory structure. You simply change your current working directory (cd command) to the dataset you are interested in running and type the name of the cmd file you wish to execute. Note that there may be several cmd files available for a particular dataset, and you can use the ls *.cmd to list them. An example cmd file (Derby.cmd) is listed below:

```
#!/bin/csh -f
#
set params = "$1 $2 " #for debug
#
# Assumes environment variables VISUAL_HOME and
# VISUAL_DATA are set appropriately
#
setenv VISUAL_DATA $VISUAL_DATA/Derby97
$VISUAL_HOME/visual $params \
        -g Derby97.ms2_i.grd \
        -i DSL120/MS_EW_RELAY.352328.2861365.2657.2147.DSL120.ss.sgi \
        -o Derby97.db \
#       -b image \
#       -b texture \
        -b sensor\
        -c none \
        -p index \
        -O .Derby97.opt \
        -W .Derby97.win
```

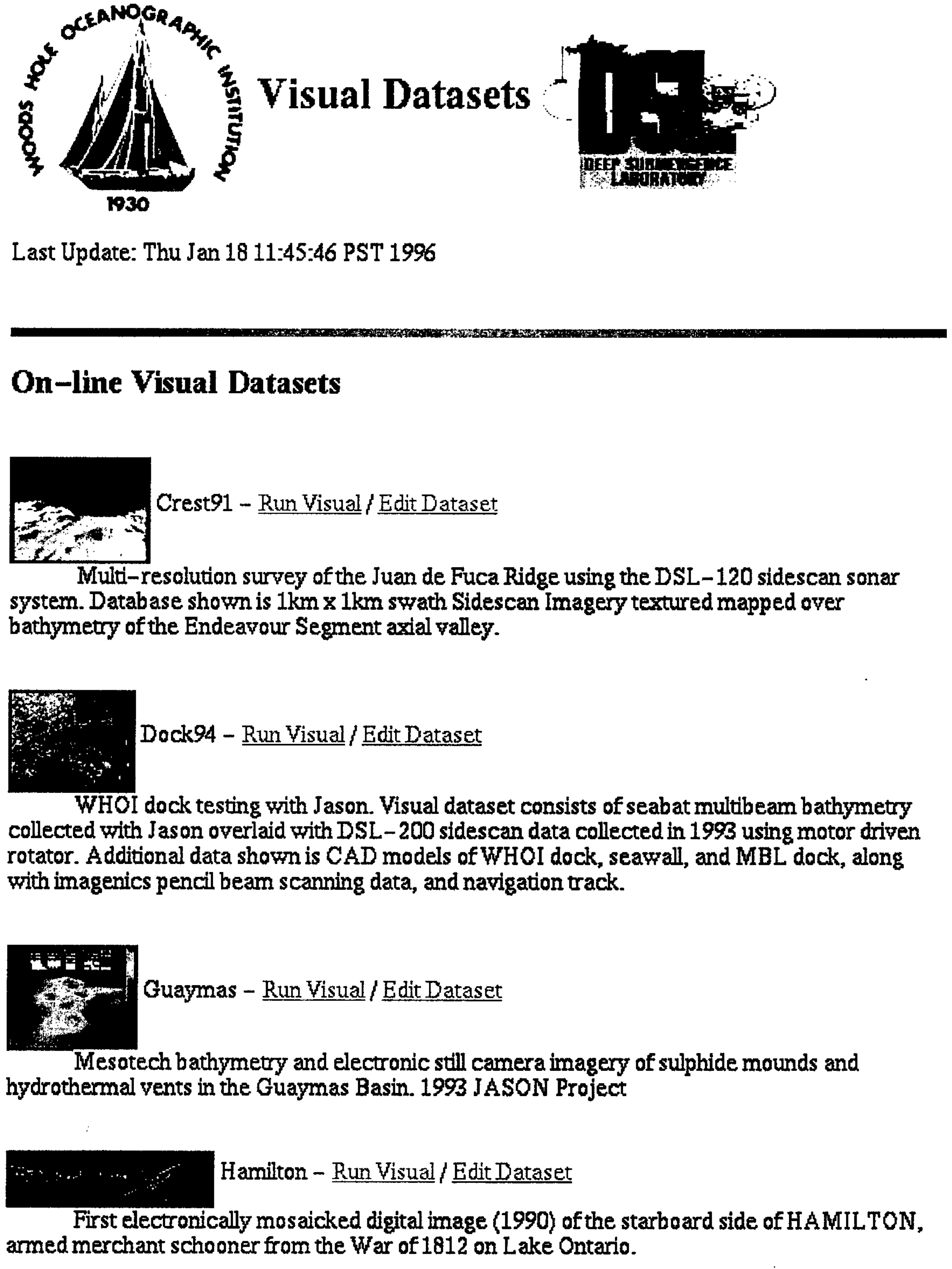
To execute the above cmd file type the following (assumes the data directory is Data/Derby97 and the cmd filename is Derby.cmd):

```
cd Data/Derby97
Derby.cmd
```

2.3 Managing Visual's Windows

Visual has several different types of windows that may appear depending on how the command file options are set when *Visual* is executed. The possible windows include the main *Visual* window, the image window, the color-map window, the button window, the sensor window, the help window, and the information window. Refer to Figure 3-2. All of these windows except the button window have a border and a title bar on top of the window. The button window is a borderless window. There are several methods for managing these windows. All of *Visual*'s windows may be managed by the standard SGI methods for managing windows. In addition, *Visual* provides a couple of alternative methods via keyboard shortcuts and the popup menu within *Visual*'s main window. The keyboard shortcuts are perhaps the most convenient way to manage the windows, but these are only available for the most common windows: Visual<V>, Buttons<b>, Info<I>, and Help<h>. All the windows may be pushed and popped via the right-button popup menu within *Visual*'s main window under *Windows* menu option. Note that any characters within <> after a menu item is the keyboard equivalent or shortcut (if available).

Note: One of the most confusing aspects of using a system like *Visual* is maintaining the appropriate cursor focus within a *Visual* window. For example, if you are in the 'pick' mode for selecting objects and have windows that overlap the main *Visual* window, you may experience some problems when the system cursor (invisible in 'pick' mode) crosses over a title bar area of an overlapping window. At that moment, the cursor focus will change from being in *Visual*'s control to the SGI window manager and during that time a red arrow will appear. The easiest way to avoid a conflict is to minimize overlapping windows. Additionally, any time that you are in 'pick' mode and want to temporarily 'freeze' *Visual*'s crosshair to get a system cursor (red arrow), simply press the 'f' key and you will have a standard SGI red arrow cursor. To resume *Visual*'s object selection operation, move the red arrow into any *Visual* window and press the 'f' key again to 'unfreeze' the crosshair and hide the red arrow cursor. Within the status window on the far upper right corner, there will be a red "Cursor Frozen" label indicating whether *Visual*'s crosshair is frozen. The crosshair will be frozen if either the 'f' key was pressed or if any objects have been selected via a left-mouse click while in *Visual*'s pick mode.



Visual Datasets

Last Update: Thu Jan 18 11:45:46 PST 1996

On-line Visual Datasets

Crest91 - Run Visual / Edit Dataset

Multi-resolution survey of the Juan de Fuca Ridge using the DSL-120 sidescan sonar system. Database shown is 1km x 1km swath Sidescan Imagery textured mapped over bathymetry of the Endeavour Segment axial valley.

Dock94 - Run Visual / Edit Dataset

WHOI dock testing with Jason. Visual dataset consists of seabat multibeam bathymetry collected with Jason overlaid with DSL-200 sidescan data collected in 1993 using motor driven rotator. Additional data shown is CAD models of WHOI dock, seawall, and MBL dock, along with imagenics pencil beam scanning data, and navigation track.

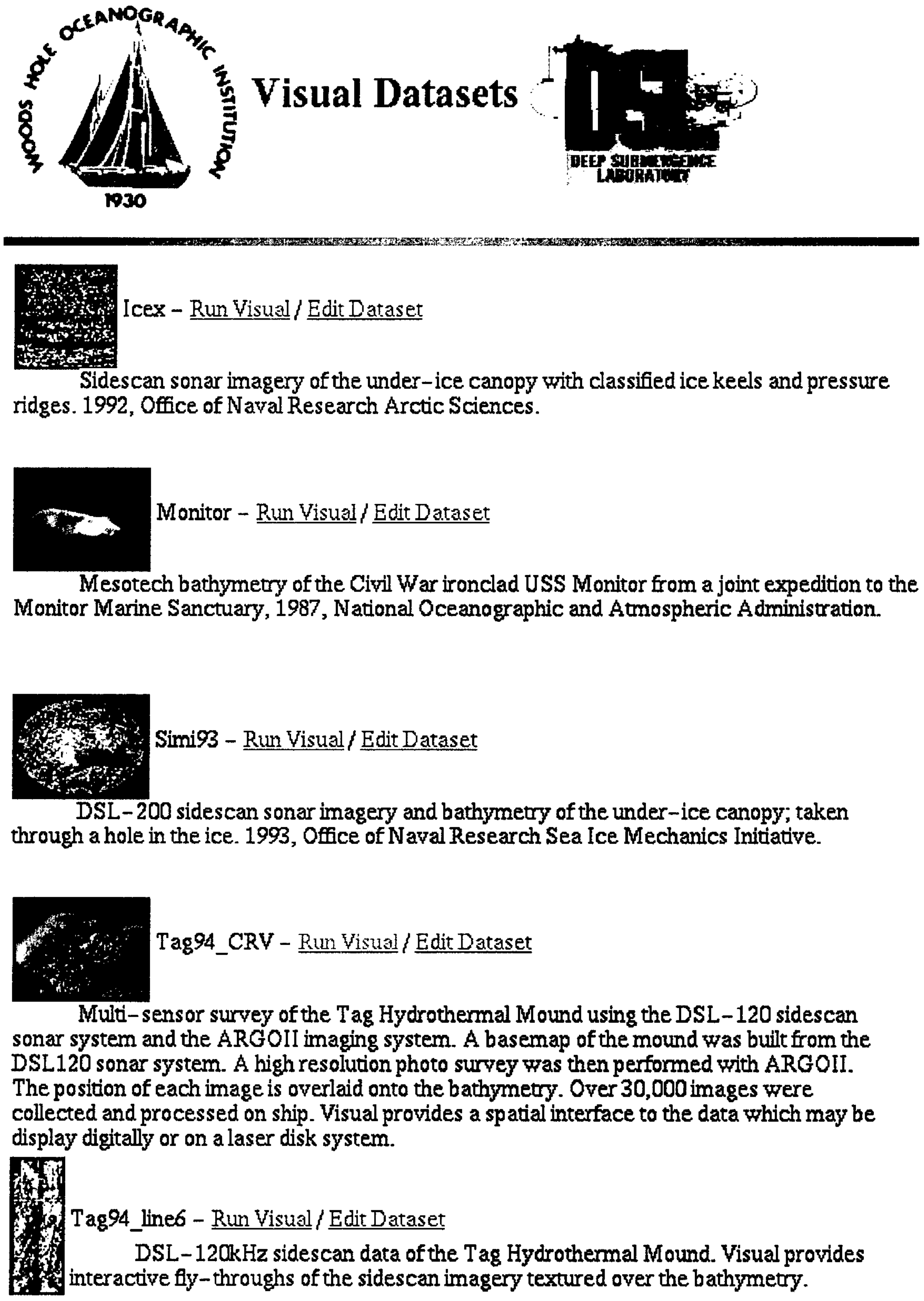
Guaymas - Run Visual / Edit Dataset

Mesotech bathymetry and electronic still camera imagery of sulphide mounds and hydrothermal vents in the Guaymas Basin. 1993 JASON Project

Hamilton - Run Visual / Edit Dataset

First electronically mosaicked digital image (1990) of the starboard side of HAMILTON, armed merchant schooner from the War of 1812 on Lake Ontario.

Figure 2-1: ***Visual*'s Netscape Interface with example datasets**



Visual Datasets

Icex – Run Visual / Edit Dataset

Sidescan sonar imagery of the under-ice canopy with classified ice keels and pressure ridges. 1992, Office of Naval Research Arctic Sciences.

Monitor – Run Visual / Edit Dataset

Mesotech bathymetry of the Civil War ironclad USS Monitor from a joint expedition to the Monitor Marine Sanctuary, 1987, National Oceanographic and Atmospheric Administration.

Simi93 – Run Visual / Edit Dataset

DSL-200 sidescan sonar imagery and bathymetry of the under-ice canopy; taken through a hole in the ice. 1993, Office of Naval Research Sea Ice Mechanics Initiative.

Tag94_CRV – Run Visual / Edit Dataset

Multi-sensor survey of the Tag Hydrothermal Mound using the DSL-120 sidescan sonar system and the ARGOII imaging system. A basemap of the mound was built from the DSL120 sonar system. A high resolution photo survey was then performed with ARGOII. The position of each image is overlaid onto the bathymetry. Over 30,000 images were collected and processed on ship. Visual provides a spatial interface to the data which may be display digitally or on a laser disk system.

Tag94_line6 – Run Visual / Edit Dataset

DSL-120kHz sidescan data of the Tag Hydrothermal Mound. Visual provides interactive fly-throughs of the sidescan imagery textured over the bathymetry.

Figure 2-2: *Visual*'s Netscape Interface with example datasets (continued)

To summarize, there are two primary methods for pushing, popping and moving windows.
Visual method (recommended):

1. Moving Windows: click and drag the top border of the window. If the window has no border, hold the Alt key down while pressing the right-mouse button over the window and select 'Move'.
2. Pushing/Popping Window: Place the cursor in any *Visual* window and press the keyboard equivalent. For example, to push/pop the info window, place the cursor in any *Visual* window and press '**i**' to pop the info window. Press '**i**' again to push the info window. Try it again with '**b**' for the button window.

SGI Standard method:

1. Moving Windows: click and drag the top border of the window. If the window has no border, hold the Alt key down while pressing the right-mouse button over the window and select 'Move'.
2. Pushing/Popping Windows: To pop a window, click on the top border of the window. Keyboard equivalents are Alt-F1 to pop a window, and Alt-F3 to push a window.

Hint: If the windows look a mess, for example if the main *Visual* window was put into the background, move the cursor to any *Visual* window and press Shift-V. This will pop the main *Visual* window (be sure to hold down the shift key until the display is fully updated). Now, to push and pop the button and info windows, press '**b**' and '**i**' respectively. Note: The cursor must be in a *Visual* window whenever you use keyboard shortcuts).

2.4 Button Window

The button window is a simple graphical user interface to common features within *Visual* (refer to Figure 2-3). The button window is a borderless window but may be moved and pushed/popped as described in the previous section. The choices in the button window are not necessarily complete and there may be other methods, such as using keyboard shortcuts, that may be more appropriate. Below is a brief description of the options found in the button window along with recommendations if there are better ways of achieving the same results.

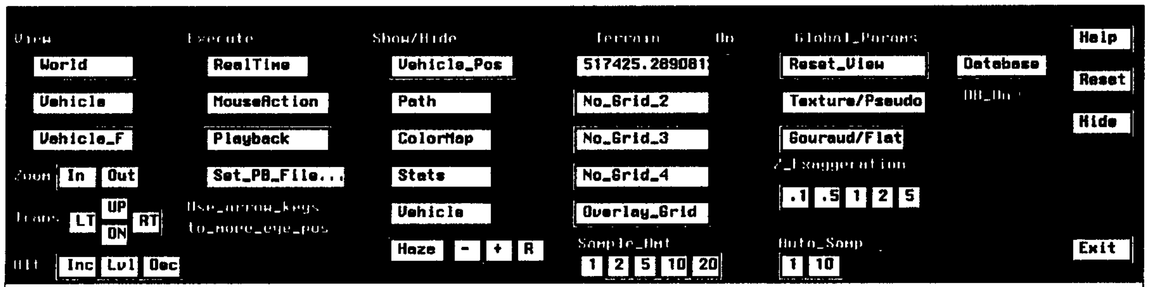


Figure 2-3: Button Window

View: Changes the viewpoint between World, Vehicle, and Vehicle_F (vehicle view with a fixed heading). The keyboard shortcut 'v' is recommended over the View options within the Button Window unless you need a vehicle view with a fixed heading which is useful if monitoring a vehicle in real-time. The v-key will toggle between World View, Plan View, and Vehicle View. Note: Plan view is currently not an option within the button window.

Zoom: These buttons will zoom-in and zoom-out to the center of the display within the main *Visual* window. You can hold down the left-mouse button for continuous zoom capability. The keyboard shortcut is the z-key and the shift Z-key for zoom-in and zoom-out respectively. You may also zoom-in by using the middle mouse button and zoom-out by using the shift-middle mouse button. The middle-mouse button will zoom-in and zoom-out to the current cursor location when in pick mode. This is convenient if you are targeting a particular location. The middle-mouse method for zooming-in and zooming-out is recommended since you don't need the button window visible.

Trans: These buttons will translate the objects within the main *Visual* display left, right, up, or down. You can hold down the left-mouse button for continuous translation capability. You may find that the arrow-keys are easier to use. Translation only works in World View or Plan View, not in Vehicle View. In Vehicle view, the shift-arrow keys will alter the orientation: use left/right arrows for heading and shift-up/down arrows for pitch. Note: flying with a heading offset is not recommended.

Alt: When in simulate mode (also know as interactive flying), these buttons will either increase, decrease, or level the vehicle's altitude. It is recommend that you use the keyboard and mouse equivalents since moving the mouse to these buttons will change the vehicle's position. In simulation mode, the middle-mouse button will increase altitude, left-mouse will decrease altitude, and a-key will level the vehicle. Note that the mouse buttons in the simulation mode will add or subtract to the altitude. For example, if you press the middle-mouse button twice, the rate of altitude change will increase. Pressing the left-mouse button once will slow that rate, and pressing the left-mouse button again will level out the altitude.

Execute:

RealTime:	Enables real-time data mode whereby *Visual* is automatically looking for real time data via the shared-memory interface. The default is enabled and this should not be changed.
MouseAction:	This is another name for simulation or interactive flying. Recommend using the s-key shortcut to toggle simulation mode on/off. When in this mode, the mouse acts like a joystick relative to the center of the main *Visual* window. If the cursor is in the center of the display, there will be no movement of the vehicle. If moved above the centerline, the vehicle will move forward. If moved below the centerline, the vehicle will move backward. Moving the cursor left and right of the centerline will cause the vehicle to rotate. Refer to Alt above for a description of using the mouse buttons to control altitude.
Playback:	Automatically playback previously recorded flights from a playback file.
Set_PB_File...:	Sets the name of the playback file.

Show/Hide:

Vehicle_Pos:	Toggles whether vehicle navigation and attitude information is shown in the information window. Useful for real-time monitoring and in simulation mode if you want to know where the vehicle is located.
Path:	Toggles whether to display a vehicle trace path in the *Visual* main window.
ColorMap:	Pushes and pops the color-map window if it was loaded during initialization.
Stats:	Displays some graphic statistics such as frame/sec in the information window.
Vehicle:	Toggles whether to display a CAD model of the vehicle (Jason) in the *Visual* main window.
Haze:	Enables and increases/decreases underwater haze effect. May obtain different results depending on the real-world scale of the dataset.

Terrain:

Lists up to four terrain grids that may be loaded simultaneously. Multiple terrain grids may be loaded and used to depict multiple resolutions or different geographical regions. Click on the name to select the terrain for choices such as overlay grid and sample_amt as these apply only to one terrain grid at a time. The toggle to the right is to show/hide the terrain.

Overlay_Grid:	Will overlay grid-lines on top of the selected terrain
Sample_Amt:	Will sub-sample the selected terrain. Note: Auto_Samp overrides this selection.

Global Params:

Reset_View:	Resets view if one gets lost. If this doesn't work, recommend clicking on Reset choice at right.
Texture/Pseudo:	For datasets that support textured map terrain, this toggles the texture map on/off.
Gouraud/Flat:	For datasets that are not textured mapped or the textured is turned off, this toggles between the different polygon shading method to use, either gouraud shading or flat shading.

Z_Exaggeration: Exaggerates z-scale by selected amount.

Auto_Samp: Useful for large dataset when interactively flying through the terrain. While in simulation mode, auto_samp will sub-sample the terrain to the value on the right. When stopped (via the s-key), the terrain will progressively sharpen up to the sub-sample value on the left.

Database:

Allows one to toggle on/off each individual database entry. Click with left-mouse button to view menu. Select items within menu with the right-mouse button. To close menu, click close entry at the top with the right-mouse button.

DB_On: Toggles to show/hide all database items. This is sometimes useful for extremely large datasets where one wishes to navigate to a particular area first before turning on all the objects.

Help: Shows help window. You can push/pop the help menu with the h-key.

Reset: Useful if all else fails. Primarily used when one flys off to never-never land.

Hide: Hides the button window by pushing it. Recommend using the b-key for pushing/popping the button window.

Exit: Exits *Visual*. You can also press the 'Escape' key when the cursor is in any *Visual* window. You will be asked to confirm whether you wish to really quit *Visual*.

2.5 Info Window

The information window provides a variety of information including Status, Grid Info, Cursor Info, and Vehicle Info. Refer to Figure 2-4. Note: If the text items are too close together and are hard to read within the info window, simply expand the size of the info window by dragging one of the corners.

Figure 2-4: Information Window

A brief description of the sections within the info window follows:

Status:

View: World View, Plan View, or Vehicle View. Note: v-key toggles between each viewpoint.

Mode: The m-key toggles between modes: Pick, Measure, and View. Pick mode is used for selecting objects. The search radius is shown within [] and is in meters. Use the left-mouse button to select objects. Note: as you drag the crosshair over objects, the information will be displayed in the Cursor info section. In measure mode, two crosshair cursors are displayed for measuring objects. Use the left-mouse button to alternate moving each of the two cursors.

Mouse: Displays different cursor styles.

SB: Used for spaceball input device.

Cursor Frozen: Indicates whether the cursor is frozen due to selecting objects or by pressing the f-key. Press the either the f-key or the left-mouse button while in pick-mode to unfreeze the cursor.

Grid Info:

This section displays basic grid information including origin, size, and min/max values. The viewport is the size of the area shown while in Plan View.

Cursor Info:

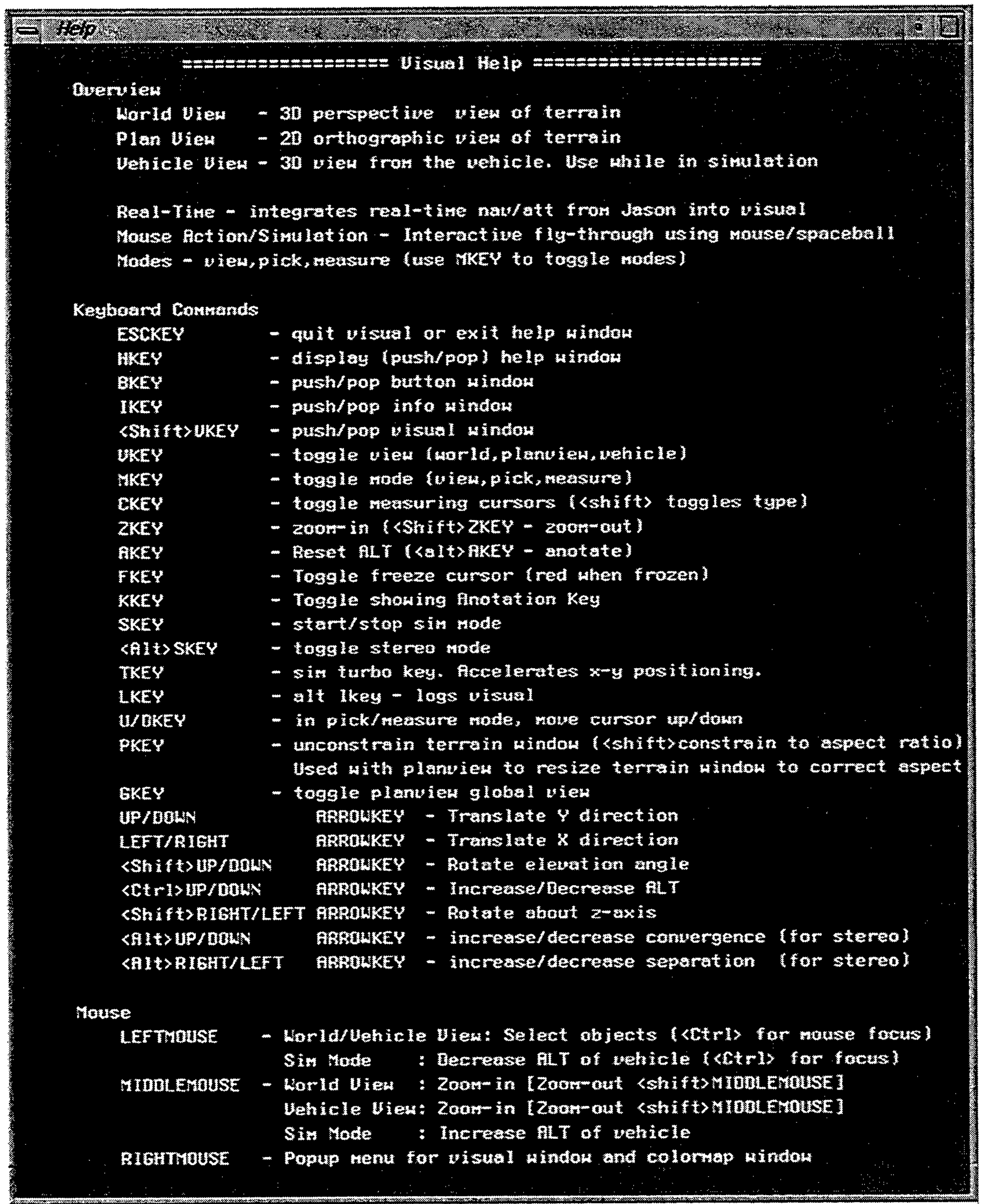
The cursor info section shows the cursor location and will display information about objects while in pick mode. In measure mode, two crosshair cursors are displayed for measuring objects. Use the left-mouse button to alternate moving each of the two cursors. Both the cursor locations and the distance between them are shown in this section.

Vehicle Info:

The vehicle info section displays the vehicle navigation and attitude information. This section may be shown or hidden via the button window.

2.6 Help Window

In addition to the help window shown below, there is on-line documentation available via the VIEW_DOC program within the VISUAL/Doc directory. To run the VIEW_DOC program, type the following:
`cd VISUAL/Doc<cr>VIEW_DOC<cr>.`



```
               ================== Visual Help ====================
     Overview
         World View   - 3D perspective  view of terrain
         Plan View    - 2D orthographic view of terrain
         Vehicle View - 3D view from the vehicle. Use while in simulation

         Real-Time - integrates real-time nav/att from Jason into visual
         Mouse Action/Simulation - Interactive fly-through using mouse/spaceball
         Modes - view,pick,measure (use MKEY to toggle modes)

     Keyboard Commands
         ESCKEY          - quit visual or exit help window
         HKEY            - display (push/pop) help window
         BKEY            - push/pop button window
         IKEY            - push/pop info window
         <Shift>VKEY     - push/pop visual window
         VKEY            - toggle view (world,planview,vehicle)
         MKEY            - toggle mode (view,pick,measure)
         CKEY            - toggle measuring cursors (<shift> toggles type)
         ZKEY            - zoom-in (<Shift>ZKEY - zoom-out)
         AKEY            - Reset ALT (<alt>AKEY - anotate)
         FKEY            - Toggle freeze cursor (red when frozen)
         KKEY            - Toggle showing Anotation Key
         SKEY            - start/stop sim mode
         <Alt>SKEY       - toggle stereo mode
         TKEY            - sim turbo key. Accelerates x-y positioning.
         LKEY            - alt lkey - logs visual
         U/DKEY          - in pick/measure mode, move cursor up/down
         PKEY            - unconstrain terrain window (<shift>constrain to aspect ratio)
                           Used with planview to resize terrain window to correct aspect
         GKEY            - toggle planview global view
         UP/DOWN             ARROWKEY  - Translate Y direction
         LEFT/RIGHT          ARROWKEY  - Translate X direction
         <Shift>UP/DOWN      ARROWKEY  - Rotate elevation angle
         <Ctrl>UP/DOWN       ARROWKEY  - Increase/Decrease ALT
         <Shift>RIGHT/LEFT ARROWKEY  - Rotate about z-axis
         <Alt>UP/DOWN        ARROWKEY  - increase/decrease convergence (for stereo)
         <Alt>RIGHT/LEFT     ARROWKEY  - increase/decrease separation  (for stereo)

     Mouse
         LEFTMOUSE      - World/Vehicle View: Select objects (<Ctrl> for mouse focus)
                          Sim Mode     : Decrease ALT of vehicle (<Ctrl> for focus)
         MIDDLEMOUSE    - World View   : Zoom-in [Zoom-out <shift>MIDDLEMOUSE]
                          Vehicle View: Zoom-in [Zoom-out <shift>MIDDLEMOUSE]
                          Sim Mode     : Increase ALT of vehicle
         RIGHTMOUSE     - Popup menu for visual window and colormap window
```

Figure 2-5: Help Window

2.7 Common User Operations (2D/3D)

2.7.1 Toggle Views

There are three available viewpoints within *Visual*: World View, Plan View, and Vehicle View. The v-key will toggle between these views and the info window will display the currently active view within the status line. For selecting objects, Plan View is the most useful. Note: when handling large datasets, *Visual*'s performance is greatly enhanced when zoomed-in to a small Plan View area since only the objects within the display area are drawn and available for selection.

2.7.2 Panning/Zooming

There are several ways to pan and zoom within *Visual*. The recommended approach for navigating for object selection is to be in Plan View and in Pick Mode. Use the middle-mouse button to zoom-in and the shift-middle-mouse button to zoom-out. Pan left, right, up, or down by moving the crosshair near the edge of the *Visual* main window. To keep the cursor in the center while panning, press and hold the alt-key while moving the mouse. The alt-key will snap the crosshair and the view to the center of the display. Note: arrow keys may also be used for panning.

2.7.3 Selecting Objects

Use the m-key to change between modes: Pick, Measure, and View. The current mode is shown on the status line within the info window. As mentioned in the Panning/Zooming section above, the recommended approach for navigating for object selection is to be in Plan View and in Pick Mode. Use the middle-mouse button to zoom-in and the shift-middle-mouse button to zoom-out. Pan left, right, up, or down by moving the crosshair near the edge of the *Visual* main window. To keep the cursor in the center while panning, press and hold the alt-key while moving the mouse. The alt-key will snap the crosshair and the view to the center of the display.

As you drag the crosshair over objects, information as to how many objects are in the search radius and information about those objects are displayed in the info window. Use the plus-key and minus-key to increase and decrease the search radius. The size of the search radius is displayed in the status line of the info window. When the crosshair is over the object or objects of interest, use the left-mouse button to select the objects. When the objects are selected, the crosshair will automatically be frozen at the location of the selected objects and the standard SGI red arrow will appear. At this point several things can happen depending on the type of objects selected. If the objects are images, an image viewer may appear. What happens is dependent on the do_selection script, the do_selection preference file, and the type of objects that are selected.

To unfreeze the crosshair and select other objects, move the red arrow into the main *Visual* window and click the left-mouse button. The crosshair will now be unfrozen and the standard red arrow cursor should disappear. To select objects again, position the crosshair over the objects of interest and press the left-mouse button.

2.7.4 Measuring Objects

Use the m-key to change between modes: Pick, Measure, and View. The current mode is shown on the status line in the info window. The measure mode displays two crosshair cursors that may be used to do a limited amount of measuring. You can alternate moving each crosshair cursor by clicking the left-mouse button. The cursor locations, distance between them, and bounding areas are displayed in the info window.

2.7.5 Annotating

Visual includes a limited capability to manually add textual annotations. To make annotations, set the viewpoint to Plan View and the Mode to Pick. Move the crosshair over the location of the annotation. Press alt-a to make the annotation. A text input prompt string will appear with the text Annotation #01:. Enter the annotation string and press carriage-return. The number for that annotation will appear in the location selected. Pressing the k-key will display an annotation key in the bottom-right part of the main *Visual* window. Press the k-key again to hide it. To display the annotation text instead of the number, use the main popup menu 'Annotate->Toggle Show Text'. Annotations will be saved in a file called anotate.dat in the directory that *Visual* was executed from. You can modify the dataset's object database if you wish to preload these annotations the next time *Visual* is run.

2.7.6 Interactive Flying (Simulation Mode)

For interactive flying, it is recommend that you are in Vehicle View and that you use the s-key shortcut to toggle simulation mode (interactive flying) on/off. When in this mode, the mouse acts like a joystick relative to the center of the main *Visual* window. If the cursor is in the center of the display, there will be no movement of the vehicle. If moved above the centerline, the vehicle will move forward. If moved below the centerline, the vehicle will move backward. Moving the cursor left and right of the centerline will cause the vehicle to rotate. To control altitude, use the middle-mouse button to increase altitude, the left-mouse to decrease altitude, and a-key to level the vehicle altitude. Note that the mouse buttons in the simulation mode will add or subtract to the rate altitude change. For example, if you press the middle-mouse button twice, the rate of altitude change will increase. Pressing the left-mouse button once will slow that rate, and pressing the left-mouse button again will level out the altitude. You can always hit the a-key to instantly level-out the altitude. Note: holding the shift key down while flying will constrain motion to 1-axis at a time.

2.7.7 Real-Time Viewing

A standard setup for real-time viewing is difficult since it depends on what you are interested in viewing. Two-dimensional viewing of real-time observations is straightforward using the Plan View and info window. For three-dimensional viewing, the most common setup would be Vehicle View with a fixed heading via the button window. Then you would want to zoom-out using the shift z-key until you can see the vehicle CAD model in view. Use the shift up/down arrows to adjust the look-up/down angle. If you are interested in a trace of the vehicle path and vehicle information, you should selected Path and Vehicle_Pos from the button window. The vehicle navigation and attitude information is displayed in the info window. You can reset the vehicle path by using the Reset Path option within the main popup menu of the *Visual* window.

2.8 Main Popup Menu Items

The main popup menu items are available via a popup menu that is accessed by holding down the shift-right mouse button while the cursor is in the main *Visual* window. Most of these items are now available via the button window or keyboard shortcuts, which are the preferred methods. These are described elsewhere. The popup menu is shown in Figure 2-6. Note: keyboard shortcuts are within <>. Menu items containing an arrow on the right have additional menu choices.

There are still some items that are only available via the main popup menu and these are briefly described below:

Spin	- originally used for benchmarking rendering performance.
PNS	- used to record/playback interactive flights.
Show/Hide	- a few more items to show/hide.
Terrain	- a few more options for terrain rendering.
Set Mouse	- mouse options for selecting cursors.
Set SpaceBall	- spaceball options for selecting cursors.
Set Cursor Type	- sets different types of cursors.
Benchmark	- originally used for benchmarking rendering performance.
Stereo	- toggles putting monitor into stereo mode.
Annotate	- a few more options for annotation.
Insert Data	- interactive object insertion.
Snapshot	- *Visual* screen snapshot options.

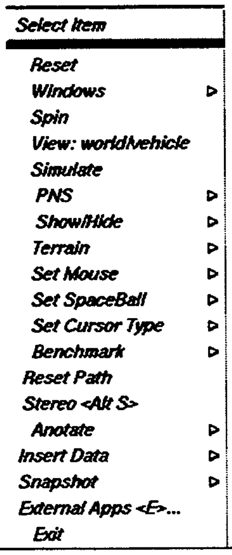


Figure 2-6: Visual main window Popup Menu

2.9 Keyboard Shortcuts

There are several keyboard shortcuts that are available. Refer to section 2.6 Help Window. Note: One keyboard shortcut not listed in the Help window is the e-key used to access external applications. See next section.

2.10 External Applications

2.10.1 External Applications Menu

External applications widely expand *Visual*'s capabilities. Most of the time these are transparent to the user since they are automatically executed via the do_selection script. External applications range from image viewers, to data editors, to custom hardware applications such as the CRV Control Application. The main external applications menu may be accessed via the e-key or via the main *Visual* popup menu. Refer to Figure 2-7.

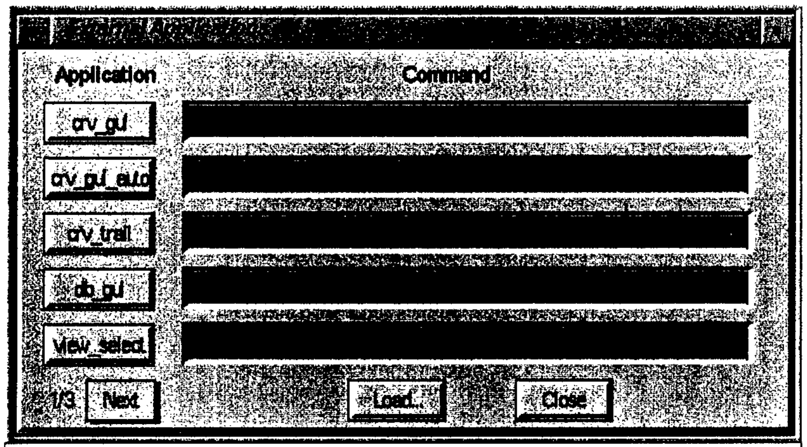


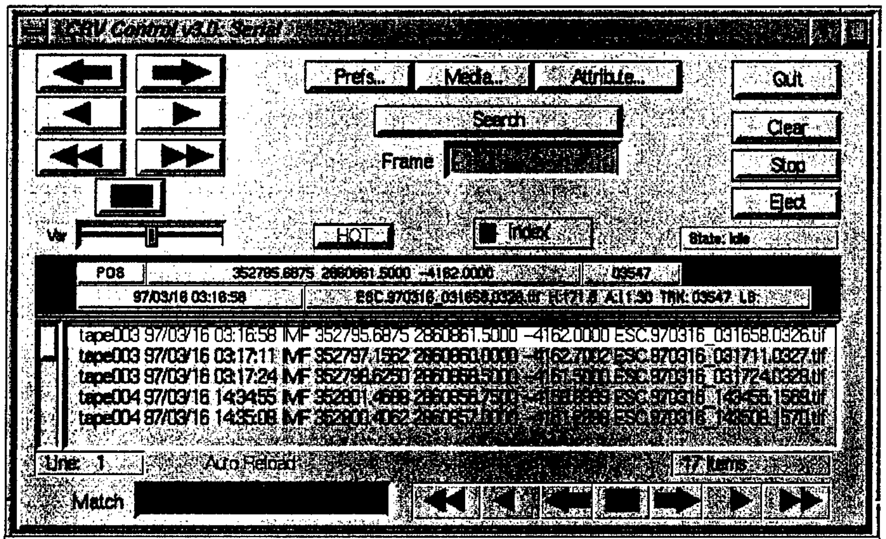
Figure 2-7: External Applications Menu

The entries of the external applications menu are easily customizable via an ASCII file (refer to section 3.5.7 External Applications Preference File for an example). A different set of application buttons and command arguments can be loaded manually within the main external applications menu via the load button.

The external applications main menu consists of pages of external applications. If the application you are interested in is not located on the current page, press 'Next'. Note: 'Next' will wrap around to the first page. To start a particular application, click on the corresponding button. Parameters to the application can be changed in the text string to the right of the application button. To quit out of the external applications main window, press 'Close'.

2.10.2 CRV Control

The CRV Control application is used to access devices such as the Sony CRV Laser Disk system or other devices that operate over a serial interface and are VLAN compatible. For this discussion, it will be assumed that a laser disk is connected to the *Visual* system via a serial interface. The CRV Control software is a fairly sophisticated application and will only be briefly discussed. Refer to Figure 2-8. The cluster of arrow buttons on the top-left of the menu control the laser disk system directly. The 'Var' slider is a variable speed control slider. As images are being displayed on the laser disk system, information about the image is displayed on the center of the control panel, and the location of the image is highlighted on the main *Visual* window. When objects are selected within *Visual*, the list in the control panel is updated with those crv accessible objects. At this time, the first image will be displayed on the laser disk monitor and the image information will be highlighted within the list of objects. To scroll through the list of selected objects, use the arrow buttons on the bottom right of the control panel or simply select a line of interest directly. A keyword pattern search may also be used on the list via the Match text input. To randomly go to a particular frame number, enter the frame number and press 'Search'. To quit the CRV Control window, press the 'Quit' button or the q-key. Note: The CRV Control Application will automatically start when objects that match the CRV criteria within the do_selection preference file are selected. When using this application as a direct interface to a laser disk or for capabilities such as the CRV Trail Application (described below), the CRV Control Application will need to be manually started via the external applications menu. The difference between the crv_gui and crv_gui_auto buttons in the external



```
POS    352795.6875 2860861.5000 -4162.0000    03547
97/03/16 03:16:58    ESC.970316_031658.0326.tif H:171.5 A:11.30 TRK: 03547 LB:
tape003 97/03/16 03:16:58 IMF 352795.6875 2860861.5000 -4162.0000 ESC.970316_031658.0326.tif
tape003 97/03/16 03:17:11 IMF 352797.1562 2860860.0000 -4162.7002 ESC.970316_031711.0327.tif
tape003 97/03/16 03:17:24 IMF 352798.6250 2860858.5000 -4161.5000 ESC.970316_031724.0328.tif
tape004 97/03/16 14:34:55 IMF 352801.4688 [illegible] ESC.970316_143455.1569.tif
tape004 97/03/16 14:35:08 IMF 352801.4062 [illegible] ESC.970316_143508.1570.tif
```

Figure 2-8: CRV Control Application

applications menu is the command arguments. crv_gui is intended for a direct connection to a CRV player without *Visual* while crv_gui_auto should be used when running *Visual* as it will automatically update the object list with the selected objects.

2.10.3 CRV Trail

When playing images back on a laser disk system, the CRV Trail Application highlights the location of the current image on the main *Visual* window in real-time. The functionality of the CRV Trail Application is now built into the CRV Control Application and for most applications it is recommended that the CRV Control Application be used instead.

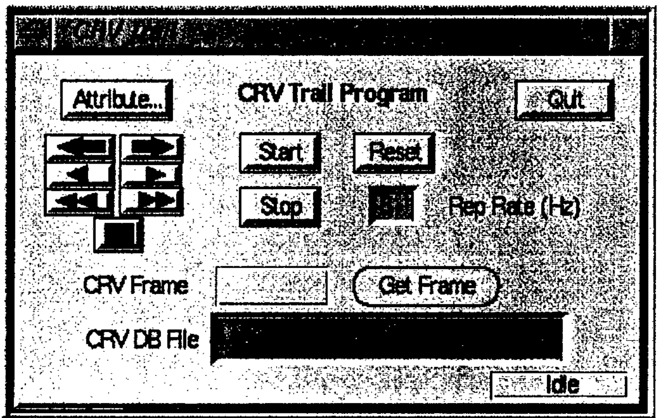
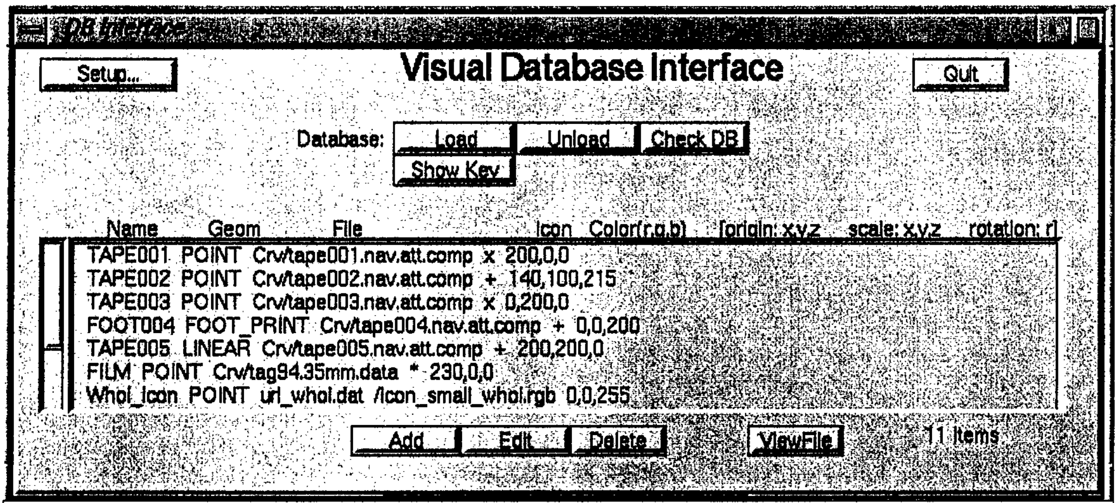


Figure 2-9: CRV Trail Application

2.10.4 DB GUI

DB_GUI is an external graphical user interface to *Visual*'s Database. DB_GUI uses either shared-memory to communicate with visual (must be run on the same computer as visual), or it uses tcp/ip sockets. You can check the current database loaded, and then add, edit, or delete a particular database entry. Refer to Figure 2-10. This application is commonly used to interactively change attributes of a database object such as setting the geometry type to point, linear, or foot_print, or changing the color of a particular object. Note:



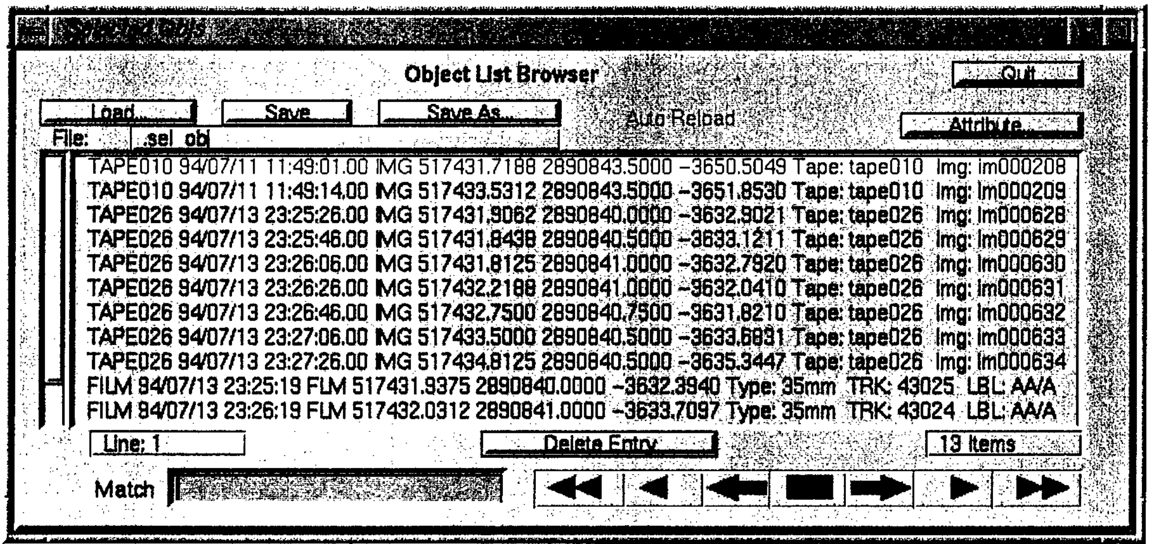
```
Name    Geom   File                                 Icon  Color(r,g,b)  [origin: x,y,z  scale: x,y,z  rotation: r]
TAPE001 POINT  Crv/tape001.nav.att.comp x 200,0,0
TAPE002 POINT  Crv/tape002.nav.att.comp + 140,100,215
TAPE003 POINT  Crv/tape003.nav.att.comp x 0,200,0
FOOT004 FOOT_PRINT Crv/tape004.nav.att.comp + 0,0,200
TAPE005 LINEAR Crv/tape005.nav.att.comp + 200,200,0
FILM POINT Crv/tag94.35mm.data * 230,0,0
Whol_icon POINT url_whol.dat /icon_small_whol.rgb 0,0,255
```

Figure 2-10: DB_GUI Application

changes made to *Visual's* database via the DB_GUI effect only the current *Visual* session. To permanently change the database, it must be manually edited.

2.10.5 View Select

When objects are selected in *Visual*, only the number of objects and a description of one object is shown in the info window. The View Select Application displays all the objects that have been selected in a scrolling list. Refer to Figure 2-11.



```
File: .sel_obj
TAPE010 94/07/11 11:49:01.00 IMG 517431.7188 2890843.5000 -3650.5049 Tape: tape010 Img: im000208
TAPE010 94/07/11 11:49:14.00 IMG 517433.5312 2890843.5000 -3651.8530 Tape: tape010 Img: im000209
TAPE026 94/07/13 23:25:26.00 IMG 517431.9062 2890840.0000 -3632.9021 Tape: tape026 Img: im000628
TAPE026 94/07/13 23:25:46.00 IMG 517431.8438 2890840.5000 -3633.1211 Tape: tape026 Img: im000629
TAPE026 94/07/13 23:26:06.00 IMG 517431.8125 2890841.0000 -3632.7920 Tape: tape026 Img: im000630
TAPE026 94/07/13 23:26:26.00 IMG 517432.2188 2890841.0000 -3632.0410 Tape: tape026 Img: im000631
TAPE026 94/07/13 23:26:46.00 IMG 517432.7500 2890840.7500 -3631.8210 Tape: tape026 Img: im000632
TAPE026 94/07/13 23:27:06.00 IMG 517433.5000 2890840.5000 -3633.6831 Tape: tape026 Img: im000633
TAPE026 94/07/13 23:27:26.00 IMG 517434.8125 2890840.5000 -3635.3447 Tape: tape026 Img: im000634
FILM 94/07/13 23:25:19 FLM 517431.9375 2890840.0000 -3632.3940 Type: 35mm TRK: 43025 LBL: AA/A
FILM 94/07/13 23:26:19 FLM 517432.0312 2890841.0000 -3633.7097 Type: 35mm TRK: 43024 LBL: AA/A
Line: 1
```

Figure 2-11: View Select Application

Unfortunately, this application is not currently tied to any other external application. This means you can't just click on an object within this application and have an image viewer such as XV automatically be updated with that image.

2.10.6 Netscape

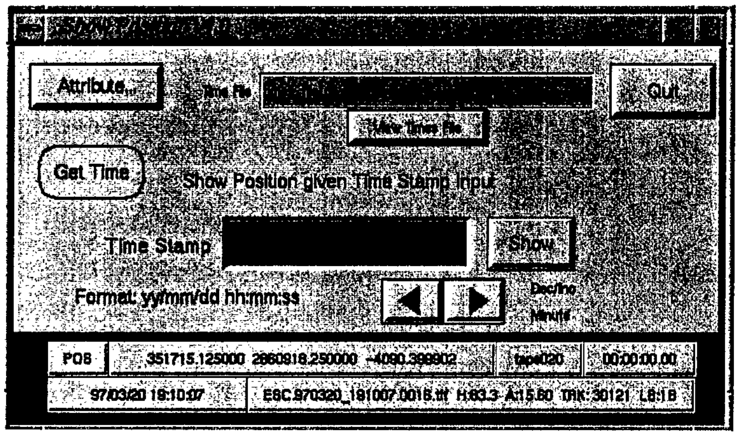
The Netscape external application is a World Wide Web (www) browser. To be used with *Visual*, it must already be running and displayed on the same monitor as *Visual* (i.e., display:0). When selecting objects with a URL type, *Visual* will update the Netscape window with the appropriate URL via the do_selection script.

2.10.7 XV

The XV external application is a generic image browser and is typically the one used when selecting image objects. *Visual*'s interface to XV is normally done via the do_selection script. If multiple objects are selected, *Visual* will display these objects within an XV window. If more than one XV window is required, it will automatically be displayed. If you want more than one window, simply de-select the objects and select them again without moving the mouse. Within XV if you wish to scroll through the images, press the space key to go forward and the backspace to go back. Make sure that the cursor is always in the XV window when issuing XV commands. To get a list of objects within XV as well as to access other XV commands, press the right-mouse button. To quit an XV window, press the q-key while the cursor is in the XV window.

2.10.8 Show Pos

The Show Pos external application is used to highlight a position within *Visual's* main window based on time. It also displays information about the object in the Show Pos display. In order to use Show Pos, a times file must be created. Refer to section 3.5.6 Show Pos.



```
POS   351715.125000 2860918.250000 -4090.399902   tape020   00:00:00.00
97/03/20 19:10:07   ESC.970320_191007.0016.tif H:63.3 A:15.60 TRK:30121 L8:18
```

Figure 2-12: Show Pos Application

2.11 Exiting Visual

There are a couple of different ways to exit *Visual*. The easiest is to position the cursor in any *Visual* window and press the 'Escape' key. Another alternative is to press 'Exit' within the button window. Finally, you can also exit *Visual* by pressing the right-mouse button with the cursor in the main *Visual* window and then selecting 'Exit' from within the popup menu. With any of the choices, you will be asked for confirmation as to whether you really want to quit *Visual* (refer to Figure 2-13). At this point, Click 'Yes' to quit *Visual* or 'No' to abort.

Hint: If you wish to save the window positions and selected options for the next time *Visual* is invoked, hold down the control-key while clicking on 'Yes'.

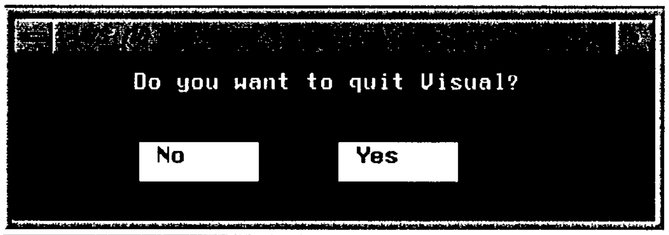

Do you want to quit Visual?

Figure 2-13: Exit Visual Confirmation Window

3 Reference Guide

3.1 Visual Architecture

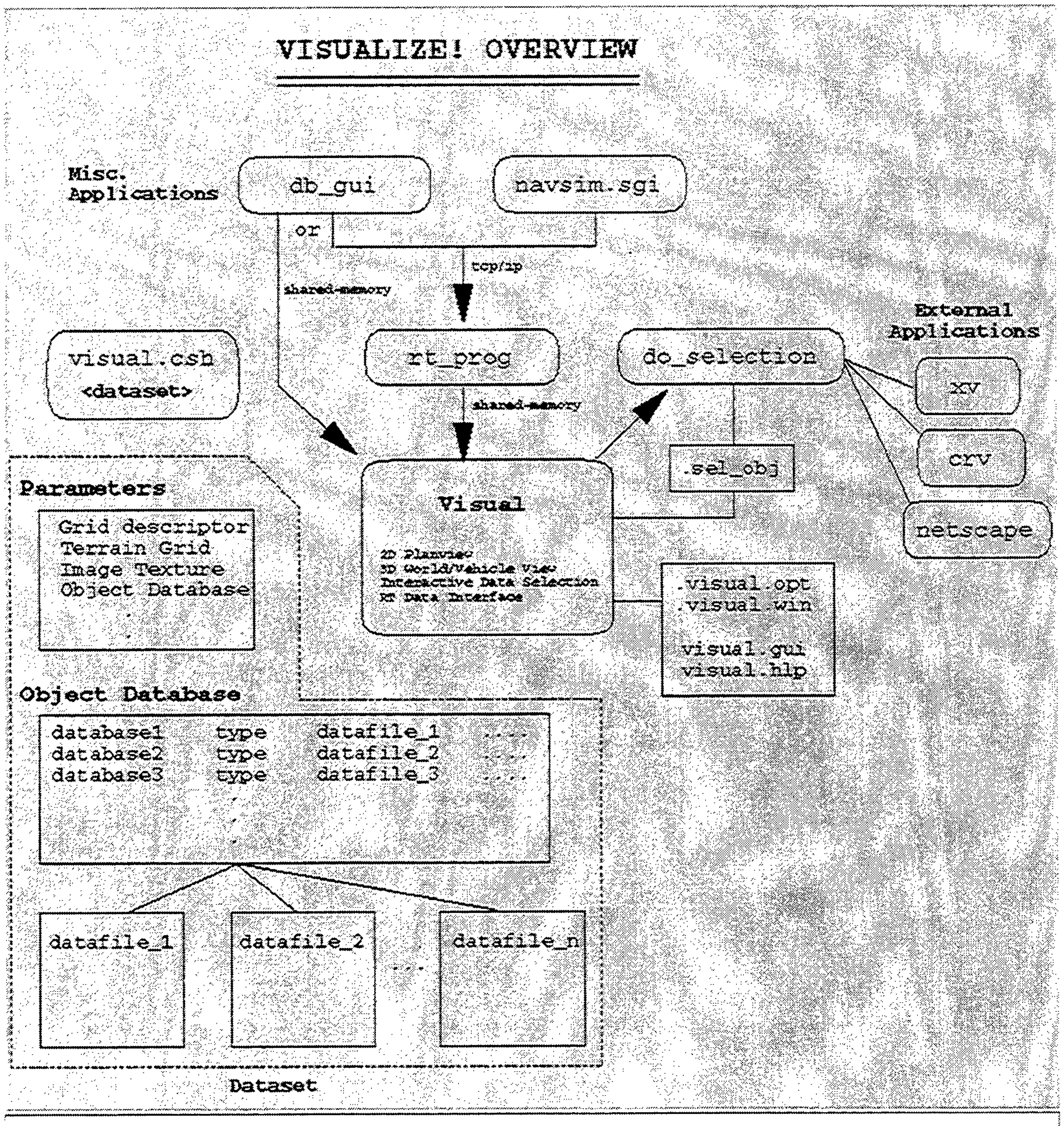


Figure 3-1: *Visual* Architecture

3.2 Visual Software

3.2.1 Visual Version/Usage

```
--- Visualize!  v2.5 ---
Written by Steve Lerner
Copyright © 1991-1999 Woods Hole Oceanographic Institution
Checking for License…OK
Usage: visual  -g grd_file
               -I img.rgb -t terr.z (ras)
               -x xorig -y yorig -z zorig
               -X xscale -Y yscale -Z zscale
               -f (sun floating pt terrain file)
               -o db_file
               -c <grid/pseudo/none> - color texture (default: grid)
               -p (full/index) - full pseudo range or color map index
               -s (stereo)
               -d (demo)
               -b <all>       - bypass all options (except terrain)
               -b <texture>   - bypass loading texture
               -b <image>     - bypass loading image & image window
               -b <sensor>    - bypass loading sensor window
               -b <terrain>   - bypass loading terrain
               -b <gui>       - bypass gui buttons
               -b <info>      - bypass info window
               -b <help>      - bypass help window
               -b <color_map> - bypass color map window
               -r no rt-time terrain redraw [default: redraw]
               -D Data Dictionary (default: data.dict)
               -O Option filename (default: .visual.opt)
               -W Win pos filename (default: .visual.win)
               -v level - verbose (err-1, wng-2, msg-3, dbg4-9)
               -h (help)

       Note: Visual Environment Variables Set
               VISUAL_HOME: /usr/people/stevel/VISUAL/Visual.v2.5
               VISUAL_DATA:  /usr/people/stevel/VISUAL/Data
```

3.2.2 C-shell program: visual.csh

visual.csh is a C shell wrapper to the visual executable. The program provides an easy interface to execute visual with a particular dataset. When new datasets have been collected or wish to be added to the list, simply edit the visual.csh program and add it. The usage of visual.csh is shown below.

```
Usage: visual.csh <name> [optional params]
       crest16    - Juan de Fuca Ridge Sidescan Imagery/Bathymetry
       dock94     - WHOI Dock94
       guaymas    - Jason Project in the Sea of Cortez
       monitor    - USS Monitor
       simi93     - Arctic - Sea Ice Mechanics Initiative
       srp93      - Mid-Atlantic Ridge
       tag94      - TAG Hydrothermal Site Sidescan Bathymetry
       tag94.ice - TAG Hydrothermal Site with ESC images
       tag94.crv - TAG Hydrothermal Site with ESC images and CRV
       tag94.shade.crv - Same as above but with shaded-relief background
       visual.doc - Visual Documentation via Netscape Browser
```

3.2.3 Visual Software Listing

```
                VISUAL/

Data/    Doc/    Src/     Visual.vn/ visual.csh*
                 /\
              Util/  Visual.vn
```

Data/ **VISUAL_DATA**

The typical dataset sub-directory structure is as follows:

Dataset_Dir

dataset.grd	Grid descriptor file
dataset.terrain.ras	Terrain data file
dataset.image.sgi	Image data file
dataset.db	Database file
dataset.data_type.data	Ascii Data

Doc/ **Documentation Directory**

VIEW_DOC*	**Script to View Documentation**
db.data_types	
db.db_file	
db.grd	
db.image	
db.rt_db_formats	
db.terrain	
external.crv	
external.crv_trail	
external.db_gui	
external.navsim_sgi	
external.netscape	
external.xv	
icon_dsl.gif	
icon_whoi.gif	
visual.2d.external.gif	
visual.2d.gif	
visual.csh_usage	
visual.do_selection	
visual.environment	
visual.hlp	
visual.html	
visual.internal	
visual.overview.gif	
visual.overview.showcase	
visual.rt_prog	
visual.software_list	
visual.usage	

Src/ **Source Code Directory**

Util/ **Utility Source Code Directory**

Install_all	Install all utilities
apps/	external applications interface
color_edit/	Internal color editor
crv_trail_v2.1/	CRV Laser Disk/Visual Interface Program
crv_v3.1/	CRV Laser Disk Interface Program
data_tools/	miscellaneous data tools
db_add_data/	Add visual database object gui
db_add_db/	Add visual db gui
db_gui/	Visual Database Graphical User Interface
do_scripts/	Scripts called from do_selection
hot_list_v1.0/	hot list application
license/	Visual license software
navsim/	tcp/ip communication test program
navtest/	tcp/ip communication test program
ok_msg/	Internal ok message program
show_pos/	Show position external application
visctl_v1.0/	External visual control interface program
vitc_read/	vitc reader application

Visual.v2.5/ **Visual Software Source Directory**

4dgifts.c	miscellaneous routine re-defined to eliminate hardware dependencies
4dgifts.h	header file
Makedepend	Generated file from make
Makefile	Usage: make visual
Makefile.deb	Usage: make visual -f Makefile.deb
Makefile.rt_prog	Usage: make rt_prog -f Makefile.rt_prog

acad.c	source file
anotate.c	source file
crt1.obj, crtn.obj	misc. object modules
data_dict.template	template file
data_dict.c	source file
datetime.h	source file
db.c	source file
db.h	source file
db_read_data.c	source file
do_selection*	Perl program - interface to external application
do_selection.csh*	C-shell program - interface to external application
do_selection.prefs.template	template file
draw_gv.c	source file
draw_objects.c	source file
errmsg.h	source file
get_view_area.c	source file
gui.c	source file
gui_button.h	source file
help.c	source file
image.c	source file
jason.dxf	autocad dxf file of jason
license.c	source file
license.dat	license datafile
make_license*	make license datafile
make_license.c	source file
make_license.cmd*	makefile
matrix.c	source file
misc_time.c	source file
mon.data	sample monitor terrain data
mon.grd	sample monitor grid descriptor file
mon.sgi	sample monitor image data
mon.shad.sgi	sample monitor image data
monitor.opt	sample monitor option file
monitor.win	sample monitor window file
options.c	source file
prompt.c	source file
protos.h	source file
read_grid.c	source file
realtime.c	source file
rt_db.formats	description of rt db interface specification
rt_prog*	rt program (provides rt data interface)
rt_prog.c	source file
rt_prog.h	source file
sensor_char.db.template	template file
sensor_char.h	source file
sensor_data.c	source file
share.c	source file
sim.c	source file
spaceball.c	source file
stereo.c	source file
surf.h	source file
terrain.c	source file
vehicle.c	source file
visual*	visual executable
visual.banner	visual banner image
visual.c	source file
visual.csh*	C-shell wrapper to visual
visual.gui	Internal visual gui buttons data file
visual.h	source file
visual.hlp	visual help file
visual_proto.h	source file
windows.c	source file
yes_no.c	source file

Visual.v2.5/ **VISUAL_HOME (Visual.v2.5)**

add_data*	add database object (gui)
add_db*	add database (gui)
apps*	external applications interface
apps.data	data file for apps

apps.data.template	template file
color_edit	internal color editor program
crv*	external application - crv command interface
crv_gui*	external application - crv gui interface
crv_trail*	external application - crv trail program
data.dict	data dictionary
data.dict.template	template file
db_gui*	external application - db gui interface
do_audio*	called from do_selection based on selected datatypes
do_movie*	called from do_selection based on selected datatypes
do_mpeg*	called from do_selection based on selected datatypes
do_netscape*	called from do_selection based on selected datatypes
do_selection*	Perl program - interface to external applications
do_selection.prefs	preference file
do_selection.prefs.template	template file
do_xv*	called from do_selection based on selected datatypes
hot_list*	external application program
jason.dxf	autocad dxf file of jason
license.dat	license datafile
new_dataset*	creates new dataset template files
ok_msg	internal ok message program
rt_prog*	rt program (provides rt data interface)
sensor_char.db	sensor characteristic database
sensor_char.db.template	template file
show_pos*	external application program
visual*	visual executable
visual.banner	visual banner image
visual.gui	Internal visual gui buttons data file
visual.hlp	visual help file
navsim*	tcp/ip network test program
vitc_reader*	vitc reader program

3.2.4 Visual Help Reference

```
================ Visual Help ==================
Overview
  World View      - 3D perspective  view of terrain
  Plan View       - 2D orthographic view of terrain
  Vehicle View    - 3D view from the vehicle. Use while in simulation

  Real-Time                   - integrates real-time nav/att from Jason into visual
  Mouse Action/Simulation     - Interactive fly-through using mouse/spaceball
  Modes                       - view,pick,measure (use MKEY to toggle modes)

Keyboard Commands
  ESCKEY          - quit visual or exit help window
  HKEY            - display (push/pop) help window
  BKEY            - push/pop button window
  IKEY            - push/pop info window
  <Shift>VKEY     - push/pop visual window
  VKEY            - toggle view (world,planview,vehicle)
  MKEY            - toggle mode (view,pick,measure)
  CKEY            - toggle measuring cursors (<shift> toggles type)
  ZKEY            - zoom-in (<Shift>ZKEY - zoom-out)
  AKEY            - Reset ALT (<alt>AKEY - anotate)
  SKEY            - start/stop sim mode
  <Alt>SKEY       - toggle stereo mode
  TKEY            - sim turbo key. Accelerates x-y positioning.
  LKEY            - alt lkey - logs visual
  U/DKEY          - in pick/measure mode, move cursor up/down
  PKEY            - unconstrain terrain window (<shift>constrain to aspect ratio)
                    Used with planview to resize terrain window to correct aspect
  EKEY            - external applications
  UP/DOWN              ARROWKEY   - Translate Y direction
  LEFT/RIGHT           ARROWKEY   - Translate X direction
  <Shift>UP/DOWN       ARROWKEY   - Rotate elevation angle
  <Ctrl>UP/DOWN        ARROWKEY   - Increase/Decrease ALT
```

<Shift>RIGHT/LEFT	ARROWKEY	- Rotate about z-axis
<Alt>UP/DOWN	ARROWKEY	- increase/decrease convergence (for stereo)
<Alt>RIGHT/LEFT	ARROWKEY	- increase/decrease separation (for stereo)

Mouse

LEFTMOUSE	- World/Vehicle View: Select objects (<Ctrl> for mouse focus)
	Sim Mode : Decrease ALT of vehicle (<Ctrl> for focus)
MIDDLEMOUSE	- World View: Zoom-in [Zoom-out <shift>MIDDLEMOUSE]
	Vehicle View: Zoom-in [Zoom-out <shift>MIDDLEMOUSE]
	Sim Mode : Increase ALT of vehicle
RIGHTMOUSE	- Popup menu for visual window and colormap window

SpaceBall

Button on ball	- toggle cursor when in measure mode
Shift key	- limits motion to 1-axis

Misc

Plus/Minus Key	- In pick/measure mode, changes cursor search radius
CTRL Key	- In sim/pick mode, pressing ctrl freezes vehicle or cursor
ALT PKey	- toggles playback with viewing transforms (default: on)

3.2.5 Visual Environment Variables

The *Visual* program uses the following environment variables. These typically are defined in the .cshrc file for the user as well as in visual.csh C-shell wrapper, however, the environment variables may be manually defined if running the visual program directly.

VISUAL_HOME	visual software directory [defaults to .]
VISUAL_DATA	data directory [defaults to .]

Note: When loading database files or the files that they reference, visual will load an absolute path name if the file begins with "/", otherwise it will load the file relative to VISUAL_DATA.

3.2.6 Visual Internal Files

Visual uses internal files (both visible and hidden). Below is a list of these files with a quick description:

.sel_obj	- list of selected objects
.visual.opt	- visual options (used at load time)
.visual.win	- visual window positions (used at load time)
do_selection	- Perl script. Provides external application interface
visual.gui	- Visual button data file
visual.hlp	- Visual help data file

3.2.7 Visual RT Program

The visual real-time program interface (rt_prog) is a standalone application which reads data from a standard tcp/ip port and shares the information with visual via shared-memory. navsim.sgi is an example of a program that can communicate to the rt_prog via tcp/ip. The rt_prog usage is as follows:

```
usage: rt_prog -p port [ -g multicast group | -i aa.bb.cc.ee ]
   -p designates the port for network packets
   -g designates the multicast group to join by name
   -i designates the multicast group to join by dot-format IP address
   -n use nav depth instead of att depth
```

-v verbose

Currently only Jason vehicle data (navigation and attitude) and visual database commands (VDB type) are supported as a direct interface to the rt_prog. The VDB commands allow all supported data types to be added, deleted, etc. Refer to RT DB Cmd Formats for a list of supported RT DB commands. Note: The main difference is that direct vehicle data is currently not stored as database objects.

3.3 Visual Database Files

3.3.1 Brief Overview

Below is a brief list of types of files that interface to/from Visual.

ASCII Grid Descriptor
: Describes background grid - origin, size, scale, etc.

Background Terrain/Image Grid
: 3D Terrain, 2D image (digitized map, mosaic, blueprint, etc.)

ASCII Database
: datafile, geometry, icon, color, etc.

ASCII Datafiles
: Type yy/mm/dd hh:mm:ss.s [type dependent data]

ASCII Data Dictionary
: Eliminate code change to import different data types

ASCII do_selection preference file
: Maps data type to specified application when selected

3.3.2 Dataset Descriptor File

The dataset file (.ds extension) contains information describing where to find different datafiles for a particular dataset. The file consist of attribute value pairs deliminated by a colon. This file primarily is used to build the dataset html file; used when viewing the database via a WWW browser such as netscape. Here is an example of a dataset file.

```
File: Dock94.ds

        Dataset_name:  Dock94
        Description:   Dock94.desc
        Visual_cmd:    Dock94.cmd
        Icon_small:    Dock94.icon
        Icon_large:    Dock94.large.icon
        HTML_file:     Dock94.html
```

3.3.3 Grid Descriptor File

The grid descriptor file is an ASCII file which describes the grid dataset. Below is a sample grid descriptor file. Note: Some of the parameters can be overridden as command line arguments to visual.

```
Grd.version:          DD_GRD_1.0
Grd.creator:          xyz2grd V3.0
Grd.creationTime:       Jul 26 1994 14:51:45
Grd.source:             ./line6.srf.xyz
Grd.sourceTime:         Jul 25 1994 14:29:50
Grd.sensorClass:        Sonar
Grd.unit:               Meter
Grd.precision:          2
Grd.nullValue:          0.0
Grd.xMinimumValueFound:         515257.125000
Grd.xMaximumValueFound:         517491.687500
Grd.yMinimumValueFound:         2886461.500000
Grd.yMaximumValueFound:         2890260.500000
Grd.zMinimumValueFound:         3579.46
Grd.zMaximumValueFound:         3799.82
Grd.projectionType:     XY
Grd.coordinateUnit:     Meter
Grd.coordinateSense:    Left-Hand
Grd.coordinateLattice:  Lattice
Grd.xOrigin:            516585.000000
Grd.yOrigin:            2890050.000000
Grd.zOrigin:            0.000000
Grd.xScale:             2.000000
Grd.yScale:             2.000000
Grd.zScale:             1.000000
Grd.zRotation:          0.0
Grd.format:             Sun Raster
Grd.screen:             True
Grd.type:               Single
Grd.depth:              32
Grd.dimension:          1
Grd.order:              Row-Major
Grd.xSize:              750
Grd.ySize:              1250
Grd.headerLength:       800
Grd.location:           External
Grd.fileName:           tag94.line6.srf.ras
Grd.filePath:           .
```

3.3.4 Image File

The image file is a binary SGI-formatted rgb image file which is typically used as a texture that overlays onto the 3D bathymetry.

3.3.5 Terrain File

The terrain file is a sun raster formatted data file. Currently both standard (8-bit) and custom (32-bit) sun floating-point raster file formats are supported. The terrain file provides the 3D information used typically for bathymetry.

3.3.6 Object Database File

The object database is a meta database file for visual. It is an ASCII file containing entries with a database name, type, datafile, etc. Comments begin with a # in the first column. A sample database is shown below. Note: If the datafile begins with a "/" then an absolute data path is used, otherwise it is assumed relative to the environment variable VISUAL_DATA.

```
# Test database file for 3D Visual
#
# Name : Name used in database menu
# Type : point, linear, poly_obj, anotation, foot_print, acad
# Icon : point   '.'         specifies dot
#                '+' or 'x'  specifies 1 unit size icon (plus or x)
#               '/'<file>   specifies icon is an icon_file (rgb)
#                string      specifies string icon
#        linear  '-'         specifies no icon
#               string     specifies string icon
#       poly_obj   N/A
#       anotation  N/A
#       foot_print N/A
# Color: r,g,b (N/A for poly_obj)
# [Origin: x,y,z  Scale:  x,y,z  Rotate: r] (Default: 0.,0.,0. 1.,1.,1. 0.)
#
#     Name      Type    Datafile                           Icon    Color(r,g,b)  [origin
scale    rotate]
      WHOI_DOCK poly_obj  /absolute/path/whoi_dock.obj        -     170,170,0
0.,0.,1.2   -1.,-1.,1.    -55.
      ESC_TAPE1  point    tape001.esc.nav.att.comp            x     200,0,0
      TAPE002    point    tape002.esc.nav.att.comp            +     200,0,0
      TAPE003    point    tape003.esc.nav.att.comp            x     0,200,0
      TAPE004    point    tape004.esc.nav.att.comp            +     0,0,200
      TAPE005    point    tape005.esc.nav.att.comp            +     200,200,0
      TAPE006    point    tape006.esc.nav.att.comp            +     0,200,200
      TAPE007    point    tape007.esc.nav.att.comp            +     200,0,200
      TAPE008    point    tape008.esc.nav.att.comp            +     200,130,180
      TAPE009    point    tape009.esc.nav.att.comp            +     100,130,180
      FILM       point    tag94.35mm.data                   · 35mm  230,0,0
      Whoi_Home  point    url_whoi.dat                       WHOI   0,0,255
      Whoi_Home  point    url_whoi.dat                 /whoi_icon.rgb 0,0,255
      DSL_Home   point    url_dsl.dat                        DSL    50,0,200
#     TAPE010    point   /net/tape010.esc.nav.att.comp        +     200,130,180
```

3.3.7 Data Dictionary

```
# Data Dictionary for Visual - SL 3/96
#
#   This files describes data formats which are not built-in to
#   Visual. Using this file, Visual knows which fields (columns) to
#   extract position information (and attitude information, if available).
#   The assumption is that the input data format is an ascii record with
#   the following format:
#       Type yy/mm/dd hh:mm:ss.s [type dependent data]
#
#   Note: The generic ascii format for Visual is as follows. Data
#          in this format need not be defined in this file unless
#          there is a specific output string required.
#       Type yy/mm/dd hh:mm:ss.s xpos ypos zpos [type dependent data]
#
#
# Data Dictionary Format
#
#   Data  --------------- INPUT -------------------------   ----- OUTPUT -----
#   Type  xpos ypos zpos pitch roll head alt src sensor id  "String: #col ..."
#         col  col  col  col   col  col  col col col    col
#   Where col is the column of input data that contains that data element.
#   Note: Use negative sign to flip sign of field, 0 for Not Available
#
#   The output string is optional. If not specified, no data will
#   be appended to selected objects and the format will be
#      DB_name  Date Time Type xpos ypos zpos
#   If the output string is specified, the selected objects format is
#   the same as above but with the output string appended.
#
# Example: PNS type data with x,y,z data in columns 7,8,9. Since
#          column 9 is depth in PNS record, use negative sign for zpos.
#          Must pad out remaining fields with 0, since not available.
#    PNS   7 8 -9 0 0 0 0 0 0 0
#
#
# Internally Defined Types: ACAD, ANO, POS, POLY, VID
#
# Data Dictionary Follows
# Data ------- INPUT ---------------    ----- OUTPUT -----
# Type x  y   z  p  r  h  a sr se id   "String: #col ..."
#
PNS    7  8  -9  0  0  0  0  0  0  0   "this is a test #7 and #8"
PAS    0  0   0  7  8  6  0  0  0  0   #gyro-5,compass-6
ICE    7  8 -12  0  0 13 11  0  0  0   "Tape: #5  Img: #6"
IMG   10 11 -15  0  0 16 14  1  9  0   "Tape: #4  Img: #5  TRK: #7 LBL: #6"
FLM    5  6  -7  0  0  0  0  0  0  0   "Type: #4  TRK: #8  LBL: #9"
URL    5  6  -7  0  0  0  0  0  0  0   "URL: #8"
MPG    5  6  -7  0  0  0  0  0  0  0   "FILE: #8"
AUD    5  6  -7  0  0  0  0  0  0  0   "FILE: #8"
CVS    4  5  -6  0  0  0  0  0  0  0
IME    4  5  -6  0  0  0  0  0  0  0
SRE    4  5   6  0  0  0  0  0  0  0
NAS    5  6  -7 11 12 10  0  0  0  0   # Nav and att
MRG    7  8  -9 12 13 11  0  0  0  0   # combined nav & att
```

3.3.8 Supported Data Types

All the supported data types for visual are ASCII files with the following generic format (TYPE is usually 3 char abbreviation or mnemonic).

```
TYPE yy/mm/dd hh:mm:ss.s [Type dependent data]
```

Below is a current list of visual supported data types. Unless otherwise specified, the data types are supported during initialization as well as in a real-time environment (RT).

```
ANO - Anotation (visual internally defined)
CVS - hacked for Dana - waypts
DAT - was temperature data (RT mode only. Should be generic sensor data)
FLM - Film
ICE - Image Camera Event
IME - Imagenics Event (scanning sonar)
IMG - Image Event - CRV interface
PAS - Attitude (currently RT mode only)
PNS - Navigation
POLY - Polygon Model (visual internally defined)
POS - generic position
SRE - Mesotech (scanning sonar)
URL - URL (Universal Resource Locator)
VID - Video
VDB - Visual Data Base (currently RT mode only)
```

<u>Data Format Examples</u>

```
ANO 95/04/19 20:30:35.64 1. 118.717590 186.803528 233.000000 Bicycle
FLM 94/07/10 07:04:01.00 35mm 517739.345836 2890644.595836 3428.87 43443 AA/A
ICE 94/07/10 21:13:12.00 ARG tape006 im000010 517715.356915 2890821.864915  -2.206896
1.300000 14.531035 3656.765000 206.431035 202.362069
IMG 94/07/09 06:18:01.00 tape001 im000014 AA/A 00032 001 TAG 517141.002150 2890771.368508
2.224138 -10.462069 12.724138 3659.573000 239.510345 238.200000
PAS 94/10/23 16:59:55.39 JAS  35.35 6.98 -11.30 -3.92 0.0 3.152 -241.3 -15.0 -0.4318
0.1758 0.0 0.0
PNS 94/10/23 16:59:55.34 NAV XYZ JS0 17.82 34.35 1.32 0.0 00
POS 94/07/10 21:10:29.00 517734.1202 2890819.1752 3656.8402  TRK: 4070  tape: tape006
image: im000001
SRE 94/11/00 00:00:00.00  10.60  -5.12 -2.0
URL 94/07/10 07:04:39.00 url 517534.090000 2890997.000000 3706.74 http://www.dsl.whoi.edu
VID 94/07/10 07:04:39.00 src 517534.090000 2890997.000000 3706.74 94/07/10 07:10:39.00
517534.090000 2891997.000000 3706.74 file:/datafile.mpg
VDB - see RT DB Cmd Formats


Sample POLY Format
------------------
POLY 94/10/23 11:10:20 WHOI_DOCK_PILE
```

```
#
#Polygon Object File Format
#  POLY  yy/mm/dd hh:mm:ss Tag
#  Label  num_vertices   r,g,b
#  vx,vy,vz
#
Piling001_side1  4 75,75,75
26.65,35.8,0.
26.65,35.8,-20.
25.95,35.8,-20.
25.95,35.8,0.
#
Piling001_side2  4 75,75,75
25.95,35.8,0.
25.95,35.8,-20.
25.95,34.8,-20.
25.95,34.8,0.
#
.
.
.
```

3.3.9 RT DB Cmd Formats

```
# Supported RT DB cmds: load, add, delete, replace, add_obj, check_db
# Generic fmt: VDB yy/mm/dd hh:mm:ss.s <CMD> [params...]
#               Parameters for each cmd type:
#                            load <file>
#                            add <DB_NAME> <geom> <file> <icon> <r,g,b>
#                            delete <DB_NAME>
#                            replace <DB_NAME> <geom> <file> <icon> <r,g,b>
#                            add_obj <DB_NAME> <tag> <date> <time> <tag dep data>
#                            check_db <file>
#
# Examples follow:
VDB 95/20/01 14:42:49.1 load    /net/Data/Tag94/tag94.db
VDB 95/20/01 14:42:49.1 add     NEW_TAPE008 point  /net/Data/Tag94/Esc/esc.fixed.tape008
+ 100,200,0
VDB 95/20/01 14:42:49.1 delete  NEW_TAPE008
VDB 95/20/01 14:42:49.1 replace NEW_DB point foo.tape008 + 100,200,0
VDB 95/20/01 14:42:49.1 add_obj NEW_DB ICE  95/20/01 14:42:49.1 517714.788597
2890747.359123  3651.401655  tape: tape008 img: im000001
VDB 95/20/01 14:42:49.1 check_db /tmp/visual.5805.db
```

3.3.10 Sensor Char DB File

```
# Sensor Characteristic DBfile for Visual - SL 11/96
#
#   This files describes sensor characteristics for data
#   loaded into Visual. The mapping between the sensors in this file
#   and the ones for a particular data entry is based on the
#   se field defined within the data dictionary.
#
#   Example:
#       The data dictionary would list the sensor column number
#       in the se column for that particular data type.
#
#      Data Entry:
#       FLM yy/mm/dd hh:mm:ss.s 35mm 100 200 2500 1.1 0.2 CAM1 25.5 10.0
#      Data Dictionary Follows
#      #Data ------- INPUT ---------------     ----- OUTPUT -----
#      #Type x  y   z  p  r  h  a sr se id   "String: #col ..."
#       FLM  5  6  -7  8  9  11 12 0 10 0
#
```

```
#       In the sensor characteristic DBfile, would contain
#       #Sensor_Name  xfov  yfov ...
#       CAM1 32.4 22.6 ...
#
#
# Sensor Characteristic DBfile Format
# Sensor_Name  xfov  yfov
TAG  32.4 22.6
```

3.4 Visual Data Tools

3.4.1 BUILD_DB_VIEW

The BUILD_DB_VIEW command (located in the document directory) generates a top-level dataset html page called visual_databases.html. This generated html page displays all the On-line Visual Datasets available.

BUILD_DB_VIEW looks for the *.ds files in sub-directories under $VISUAL_DATA/Dataset. Associated with each ds file there should be the following:

```
        Dataset.desc - description file
        Dataset.cmd  - visual cmd file
        Dataset.icon - icon (small)
        Dataset.large.icon - icon (large)
        Dataset.html - created by this program

Usage: BUILD_DB_VIEW
Output: visual_databases.html and generates Dataset.html in
        $VISUAL_DATA/Dataset directories.
```

3.4.2 New dataset

The new_dataset command, located in $VISUAL_HOME, creates a template dataset for use with Visual. The program creates the following templates (as most will need to be hand edited).

```
     name.ds   - dataset file
     name.desc - description file
     name.cmd  - visual command
     name.grd  - grid descriptor file
     name.db   - object database file

Usage: new_dataset <dataset_name>

Note: This program should be run in the data directory containing the new dataset.

Example:
-------
    % cd dataset_dir <cr>
    % new_dataset name <cr>
    Creating name.ds
    Creating name.desc
    Creating name.cmd
    Creating name.grd
    Creating name.db
    -------- Done creating Dataset ---------
       Manually edit name.desc
       Manually edit name.cmd
           Check grid,terrain,image,db files
       Manually create name.icon (GIF)

Below is an example of the templates that new_dataset creates.
----------------------------------------------------------
name.ds
    Dataset_name:   name
    Description:    name.desc
    Visual_cmd:     name.cmd
    Icon_small:     name.icon
    Icon_large:     name.large.icon
    HTML_file:      name.html

name.desc
    Dataset description goes here

name.cmd
    #!/bin/csh -f
    #
    set params = "$1 $2 " #for debug
    #
    # Assumes environment variables VISUAL_HOME and
    # VISUAL_DATA are set appropriately
    #
    setenv VISUAL_DATA $VISUAL_DATA/name
    $VISUAL_HOME/visual $params \
            -g name.grd \
            -t name.ras \
            -i name.sgi \
            -o name.db \
            -b image -b texture -b sensor\
            -c pseudo \
            -O .name.opt \
            -W .name.win

name.grd
    Grd.version:    DD_GRD_1.0
    Grd.creator:    new_dataset for Visual
    Grd.creationTime: Apr 26 1996 08:53:04
    Grd.source:     unknown
    Grd.sourceTime: Jul 25 1994 14:29:50
    Grd.sensorClass:        Sonar
    Grd.unit:       Meter
    Grd.precision:  2
    Grd.nullValue:  0.0
    Grd.xMinimumValueFound: 515257.125000
    Grd.xMaximumValueFound: 517491.687500
    Grd.yMinimumValueFound: 2886461.500000
    Grd.yMaximumValueFound: 2890260.500000
    Grd.zMinimumValueFound: 3579.46
    Grd.zMaximumValueFound: 3799.82
    Grd.projectionType:     XY
    Grd.coordinateUnit:     Meter
    Grd.coordinateSense:    Left-Hand
    Grd.coordinateLattice:  Lattice
    Grd.xOrigin:    516585.000000
    Grd.yOrigin:    2890050.000000
    Grd.zOrigin:    0.000000
    Grd.xScale:     2.000000
    Grd.yScale:     2.000000
    Grd.zScale:     1.000000
    Grd.zRotation:  0.0
    Grd.format:     Sun Raster
    Grd.screen:     True
    Grd.type:       Single
    Grd.depth:      32
    Grd.dimension:  1
    Grd.order:      Row-Major
    Grd.xSize:      750
    Grd.ySize:      1250
    Grd.headerLength:       800
    Grd.location:   External
    Grd.fileName:   name.ras
    Grd.filePath:   .
    ^L
```

```
name.db
    # Database file for 3D Visual
    #
    # Name : Name used in database menu
    # Type : point, linear, poly_obj, anotation, foot_print, acad
    # Datatfile: relative to $VISUAL_DATA unless specified absolute
    # Icon : point   '.'        specifies dot
    #                '+' or 'x' specifies 1 unit size icon (plus or x)
    #                '/'<file>  specifies icon is an icon_file (rgb)
    #                string     specifies string icon
    #        linear  '-'        specifies no icon
    #                string     specifies string icon
    #        poly_obj   N/A
    #        anotation  N/A
    #        foot_print N/A
    # Color: r,g,b (N/A for poly_obj)
    # [Origin: x,y,z  Scale:  x,y,z  Rotate: r] (Default: 0.,0.,0. 1.,1.,1. 0.)
    #
    #     Name      Type    Datafile  Icon Color(r,g,b)  [origin scale rotate]
```

3.4.3 Data Tools

```
                     Data Tools
                     ==========

Data tools are a collection of tools to aid in converting
the ASCII data files to DSL's format:

    type yy/mm/dd hh:mm:ss.s [type dependent data]
```

The tools are written in C, Perl, or csh and are located in $VISUAL_HOME/data_tools. The Usage varies among the available tools. Other unix operating system tools like grep and awk may also be used. Below is a list of available tools with a brief description and usage. These tools should be run from the data directory containing the dataset.

```
add_fields
    Descr: add textfield(s) to input file. Output to stdout.
    Usage: add_fields <filename> <deliminator> <n t> [n t]...
    where filename     - input file
          deliminator  - field separator
          n            - field number to be added
          t            - text to be added

convert
    Descr: Sample convert script - utilizes most data tools.
           Items to be change are input params: in_dir, out_dir,
          navfiles, logfiles, log_grep.
    Usage: convert

get_fields
    Descr: Extract field(s) from input file. Output to stdout.
    Usage: get_fields <filename> <deliminator> <n> [n]...
       where filename    - input file
             deliminator - field separator
             n           - field number to be extracted
join_lines
    Descr: appends lines from infile2 to infile1. Output to stdout.
    Usage: join_lines <infile1> <infile2>

jul_to_dsl
    Descr: convert julian date to dsl date. Reads from stdin.
           Output to stdout. Input file format yyyy/ddd<cr>.
    Usage: jul_to_dsl < <infile>

ll_to_utm
    Descr: Converts lat/longt to utm coords using proj.
    Usage: ll_to_utm <in_file> <out_file>
```

```
merge
    Descr: merges nav/att file with datafile based on time.
           Time format: yy/mm/dd hh:mm:ss.s, Nav x,y must be
           in column 7,8.
    Usage: merge -s source_file -m merge_file -g allowable gap
             -t type [-d]
             results to standard out
             source file is the file whose values are to be interpolated
             merge file is the file which is to be interpolated into
             gap is the allowable gap before interpolation is not performed
             type can be nav, att, or attcomp (for compass data)


merge_field
    Descr: merges field from file1 into field in file2. Output to stdout.
    Usage: merge_field <filename1> <deliminator> <n>
                       <filename2> <deliminator> <n>
        where filename1   - input file
              deliminator - field separator
              n           - field number to be read
              filename2   - merged file
              deliminator - field separator
              n           - field number to be inserted

replace_field
    Descr: replaces field in file2 from field in file1. Output to stdout.
    Usage: replace_field <filename1> <deliminator> <n>
                         <filename2> <deliminator> <n>
        where filename1   - input file
              deliminator - field separator
              n           - field number to be read
              filename2   - merged file
              deliminator - field separator
              n           - field number to be replaced

replace_fields
    Descr: replace field(s) with specified text in input file. Output to stdout.
    Usage: replace_fields <filename> <deliminator> <n t> [n t]...
    where filename    - input file
          deliminator - field separator
          n           - field number to be replaced
          t           - text to replace field

rm_fields
    Descr: Remove field from input file. Output to stdout.
    Usage: rm_fields <filename> <deliminator> <n> [n]...
        where filename    - input file
              deliminator - field separator
              n           - field number to be removed
```

3.5 Visual External Application Interface

3.5.1 Do Selection

do_selection is perl script which provides an external application interface to *Visual*. do_selection resides in VISUAL_HOME and is executed when objects are selected in *Visual* by pressing the left-mouse button while in the pick mode. If a local version of do_selection is available and executable, it will be executed instead of the one in VISUAL_HOME.

The do_selection script usage is do_selection [-p pref_file] <-f in_file>. It may be run manually for testing purposes, but it is primarily called directly from visual. The in_file is a selected objects file (*Visual* uses .sel_obj in the run directory). Note: you must have write access in the run directory for the external application interface to execute properly.

The format of the selected objects file (.sel_obj) is as follows:

```
  db_name yy:mm:dd hh:mm:ss.s Type xpos ypos zpos [type dependent data as specified in data.dict output]
```

```
Example:
TAPE009 94/07/11 07:58:57.00 IMG 517438.343750 2890795.500000 -3633.443115 tape: tape009
img: im000255 TRK: 06959 LBL: AA/A
TAPE009 94/07/11 07:59:10.00 IMG 517436.687500 2890795.750000 -3632.320068 tape: tape009
img: im000256 TRK: 06960 LBL: AA/A
TAPE009 94/07/11 07:59:23.00 IMG 517434.906250 2890795.750000 -3632.518311 tape: tape009
img: im000257 TRK: 06961 LBL: AA/A
TAPE009 94/07/11 09:09:02.00 IMG 517432.750000 2890795.750000 -3633.091064 tape: tape009
img: im000573 TRK: 07277 LBL: AA/A
```

The do_selection preference file is where the application, the data types that the application recognizes, and the cmd to run the application are defined. The do_selection script reads the preference file and then sorts out the selected objects from the .sel_obj file to individual files (.sel.app) for each application where the type from the .sel_obj file matches those listed in the preference file. The script then executes the application cmd with its individual selection file as a parameter. If there were no matches, the application cmd is not executed. Additionally, more than one application may be run simultaneously if more than one app data types matches the selected objects. The preference file default name is do_selection.prefs. If the file exists in the local directory, it will be used otherwise the one in VISUAL_HOME will be used. This is quite important as one can have custom do_selection preferences for each dataset. An example preference file is shown below.

```
#
# do_selection preference file
# To eliminate app invokation, place # in column 1.
#
# Format follows (note colon separator after app and data types)
#
# app:           data types:     cmd
crv_gui:         ICE,IMG,FLM:    $VISUAL_HOME/crv_gui -x 5 -y 216 -p /dev/ttyd2 -f
netscape:        URL:            $VISUAL_HOME/do_netscape
mpeg:            MPG:            $VISUAL_HOME/do_mpeg
xv:              ICE,IMG:        $VISUAL_HOME/do_xv
text_view:       POS:            jot -v -x
```

3.5.2 CRV Control/Trail

The CRV Contrl and CRV Trail external applications provide spatial feedback from a Sony Laser Disk System or a VLAN compatible device by displaying the location of a particular frame on *Visual*. The program reads the frame count from a Sony Laser Disk system via a direct serial link, performs a lookup through a specified crv db_file to get time and position information for the particular frame, and then provides the information to visual via shared memory. CRV Control and CRV Trail also support the VLAN232 interface. The usage for both of these programs is shown below along with the usage for a command-line interface program called crv.

```
crv_gui version 3.1
usage: crv_gui [-p port] [-f file] [-s] [-x xpos] [-y ypos] [-X size] [-Y size]
         [-m file] [-h file]
       -p <port> For example, /dev/ttya
       -f <file> to load into browser
       -s shared memory with visual
       -n DB name for visual
       -g DB geometry type for visual (linear,point)
       -i DB icon for visual (- for none)
       -c DB icon color for visual r,g,b (eg, 0,255,255)
       -a auto reload file
       -h <hot file> [default: hot_list.dat]
       -m <media definition file [type m_id port vlan(y/n) node]
       -T <crv database file> - For trail
       -r rep_rate (hZ) for trail [default: 30]
```

```
        -D <data dictionary> [default: data.dict] for trail
        -x <pos> window position
        -y <pos> window position
        -X <size> window size
        -Y <size> window size
        -v verbose
        -V VLAN232 Interface
        -d debug
```

```
crv_trail version v2.1
usage: crv_trail <-p port> [-f crv_dbfile] [-D data_dict] [-x xpos] [-y ypos]
                 [-s] [-n name] [-g geom] [-i icon] [-c color] [-o]
                 [-v] [-d] [-r rate] [-S State] [-V <-N node>]
        -p <port> For example, /dev/ttya
        -f <file> crv database file
        -D <data dictionary> [default: data.dict]
        -x <pos> window position
        -y <pos> window position
        -s shared memory with visual
        -n DB name for visual
        -g DB geometry type for visual (linear,point)
        -i DB icon for visual (- for none)
        -c DB icon color for visual r,g,b (eg, 0,255,255)
        -o output match positions to stdout
        -r rep_rate (hZ) [default: 5]
        -S state (IDLE,RUNNING) [default: IDLE]
        -v verbose
        -V VLAN232 Interface
        -N Vlan node number
       -d debug
```

```
crv version v1.0
usage: crv <-p port> {<-c cmd> | <-a cmd> | <-i>} [-v] [-d]
        -p <port> For example, /dev/ttya
        -a <cmd> ascii cmd to be sent to crv
        -c <cmd> commands: STOP,CLEAR,EJECT,STILL
                           F_PLAY,R_PLAY,F_FAST,R_FAST,F_SLOW,R_SLOW
                           F_SCAN,R-SCAN,F_STEP_STILL,R_STEP_STILL,
                           F_STEP <rate>,R_STEP <rate>,
                           SEARCH <frame>,ADDRESS_INQ,
                           INDEX_ON,INDEX_OFF
        -i interactive mode
        -v  verbose
        -d  debug
```

3.5.3 DB_GUI

DB_GUI is an external graphical *Visual* Database Interface. DB_GUI uses either shared-memory to communicate with visual (must be run on the same computer as visual), or it uses tcp/ip sockets. DB_GUI provides a graphical interface to *Visual*'s database. One can check the current database loaded, and then add, edit, or delete a particular database entry. Additionally, one can view the data file that the database name references. The usage is shown below.

```
usage: db_gui [[-s] | [-h hostname] [-p port_number]] [-f file] [-x xpos] [-y ypos]
        -s shared memory with visual
        -h <hostname>
        -p <port_number> For example, 28008
        -f <file> to load into browser
        -x <pos> window position
        -y <pos> window position
        -v verbose

Note: If shared-memory is not used, db_gui executes navsim.sgi. This executable
      must reside in the path.
```

3.5.4 Netscape

The Netscape external application is a World Wide Web (www) browser. To be used with *Visual*, it must already be running and displayed on the same display as *Visual* (i.e., display:0). Note: *Visual*'s interface to Netscape is normally done via the do_selection script.

3.5.5 XV

The xv external application is a generic image browser. *Visual*'s interface to xv is normally via the do_selection script.

3.5.6 Show Pos

The show Pos external application is used to highlight a position with *Visual*'s main window based on time. It also displays information about the object in the Show Pos display. In order to use Show Pos a times file must be created. The usage along with an example times file is shown below.

```
show_pos v1.0
usage: show_pos <-f file> [-P path] [-s] [-x xpos] [-y ypos] [-X size] [-Y size]
       -f <time file> [format: start_time end_time filename]
       -p <PATH> Path for both times file and filenames w/ times file
       -s shared memory with visual
       -n DB name for visual
       -g DB geometry type for visual (linear,point)
       -i DB icon for visual (- for none)
       -c DB icon color for visual r,g,b (eg, 0,255,255)
       -D <data dictionary> [default: data.dict] for trail
       -x <pos> window position
       -y <pos> window position
       -X <size> window size
       -Y <size> window size
       -v verbose
       -d debug
```

```
# Times file [time format yy/mm/dd hh:mm:ss]
# format: start_time end_time  filename
97/03/15 13:39:40  97/03/15 19:52:33  tape001.mrg  1578
97/03/15 19:57:18  97/03/16 02:01:07  tape002.mrg  1641
97/03/16 02:04:09  97/03/16 08:41:58  tape003.mrg  1795
97/03/16 08:45:17  97/03/16 14:44:02  tape004.mrg  1614
97/03/16 14:47:09  97/03/16 20:33:11  tape005.mrg  1562
```

3.5.7 External Application Preference File

```
#
# External apps preference file for Visual - SL 2/96
#
# Application  Command
#
crv_gui        crv_gui -p /dev/ttyd2 -x 10 -y 10 &
crv_gui_auto   crv_gui -f .sel.crv_gui -a -s -p /dev/ttyd2 &
crv_trail      crv_trail -s -p /dev/ttyd2 -f $VISUAL_DATA/Crv/tag.att.comp.overall &
db_gui         db_gui -s &
view_select    hot_list -f .sel_obj -s -a -t "Selected Objs"
view_hot       hot_list -f .sel_obj -s -a -t "Hot List"
do_pref        jot -p -800,0,-400,0 $VISUAL_HOME/do_selection.prefs
Netscape       netscape &
xv             xv &
movieplayer    movieplayer &
soundeditor    soundeditor /usr/demos/data/audio/bach.aiff &
moviemaker     moviemaker &
edit_apps      jot -p -800,0,-400,0 $VISUAL_HOME/apps.data
show_pos       show_pos -f times_file -p $VISUAL_DATA
```

3.6 Debugging Visual

The most common problems encountered when running *Visual* is that the dataset is not properly setup. Depending on the severity of the error, *Visual* will either continue with warnings or exit abnormally. The verbose flag is used to debug these problems or to simply monitor the progress of loading a dataset. *Visual* has multiple verbose levels available: 1 for errors, 2 for warnings, 3 for messages, and 4-9 for internal debugging. The verbose levels are inclusive, that is level 3 will include errors, warnings, and messages. Level 3 is recommended for most purposes. To run *Visual* with the verbose flag set, *Visual* must be either run with the command line script or the dataset command file. Simply append a –v [space] verbose-level to the command. Below is an example.

```
Derby_sample.cmd -v 3
--- Visualize!  v2.5 ---
Written by Steve Lerner
Copyright (c) 1997, Woods Hole Oceanographic Institution
Checking for License...ok
Note: Visual Environment Variables Set
        VISUAL_HOME: /usr/people/visual/VISUAL/Visual.v2.5
        VISUAL_DATA: /usr/people/visual/VISUAL/Data/Derby97

Verbose level: 3
Grid Descriptor: /usr/people/visual/VISUAL/Data/Derby97/Derby97.ms2_i.grd
Coordinate Sense: Right-Hand
xorig: 350999.5  yorig: 2860291.5  zorig: 0.0
xscale: 1.0  yscale: 1.0  zscale: -1.0
xsize: 2657  ysize: 2147
Jason Model: 142 polys

Loading data dictionary: /usr/people/visual/VISUAL/Visual.v2.5/data.dict
   defining -  PNS
   defining -  PAS
   defining -  ICE
   defining -  IMG
   defining -  FLM
   defining -  URL
   defining -  MPG
   defining -  AUD
   defining -  CVS
   defining -  IME
   defining -  SRE
   defining -  NAS
   defining -  MRG
   defining -  L
   defining -  IMF
Loading Objects from /usr/people/visual/VISUAL/Data/Derby97/DSL120/targets.dat...
Loading Object type: ANO (9 pts)
    xmin:345029.0 xmax:355094.0 ymin:2855854.0 ymax:2863841.0 zmin:0.0 zmax:4221.2
Loading Objects from /usr/people/visual/VISUAL/Data/Derby97/Crv/tape031.mrg...
Loading Object type: IMF ...
Opening sensor char dbfile: /usr/people/visual/VISUAL/Visual.v2.5/sensor_char.db
Loading Object type: IMF (1136 pts)
    xmin:351717.2 xmax:353082.6 ymin:2860501.8 ymax:2861818.8 zmin:-4285.0 zmax:-4101.6

--- Total Num data objects loaded: 1145 ---
--- xmin:345029.0 xmax:355094.0 ymin:2855854.0 ymax:2863841.0 zmin:-4285.0 zmax:4221.2 --
-

***Warning, database objects x|y min [345029.0000 2855854.0000]
    are less than grid(0) x|y origin [350999.5 2860291.5]
***Warning, database objects x|y max [355094.0000 2863841.0000]
    are greater than grid(0) max x|y [353656.5 2862438.5]
GVERSION: GL4DCRM-6.3

GVERSION: GL4DCRM-6.3

defining fog...
Loading imagefile:
VISUAL/Data/Derby97/DSL120/MS_EW_RELAY.352328.2861365.2657.2147.DSL120.ss.sgi...
Image loaded: 2660 x 2147
```

```
GVERSION: GL4DCRM-6.3

Type: 3  (2660 x 2147)
Terrain File: Not_Specified (2657x2147)

xscale: 1.0  yscale: 1.0   zscale: -1.0
xorigin: 350999.5  yorigin: 2860291.5   zorigin: 0.0

GVERSION: GL4DCRM-6.3

loading texture...
Loading option file: .Derby97.opt
```

3.7 *Visual Windows*

An example of the different types of Visual windows is shown in Figure 3-2. The image window is used to display the texture image map, if specified. The sensor window is used for real-time sensor monitoring applications. The help window (not shown below) is shown in Figure 2-5. Note: The sizes of the windows may be changed for all windows except the colormap window, sensor window, and the button window. Finally, in addition to the main popup menu, there is a popup menu in the colormap window for changing colormaps of the terrain when not using texture maps.

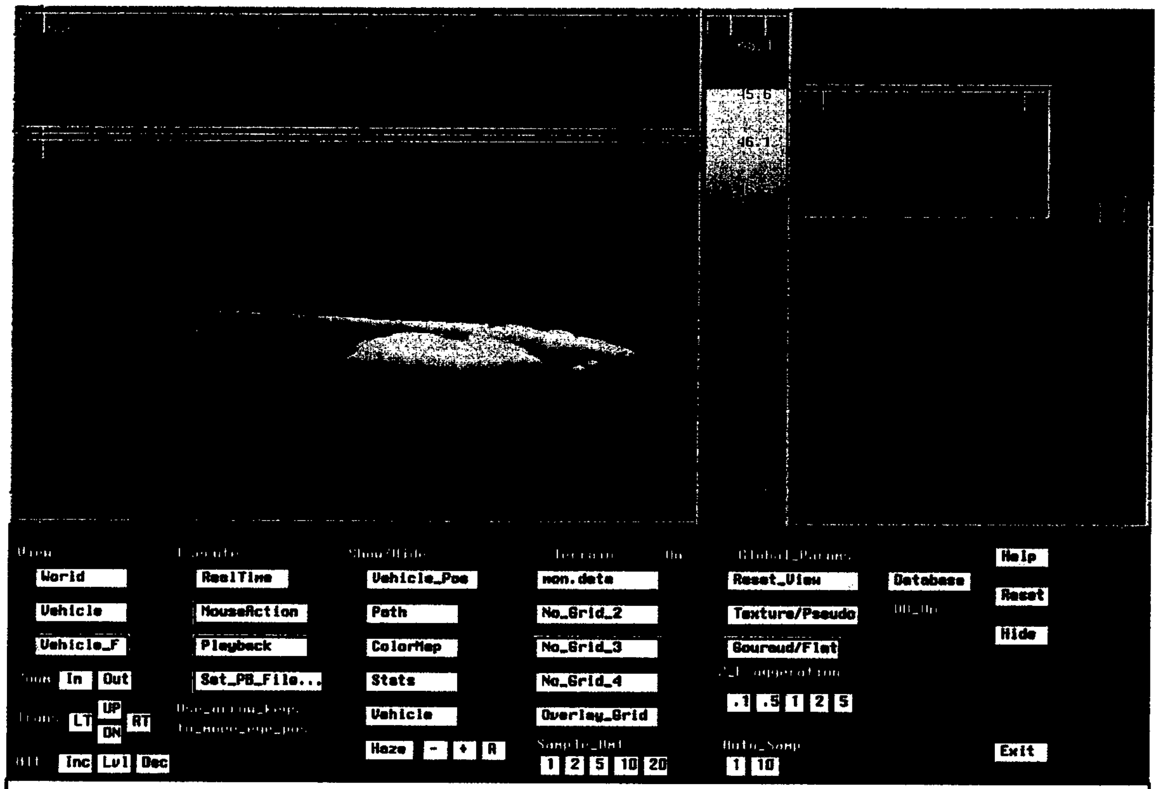


Figure 3-2: Visual Windows

Info Window (top-left), Main Visual Window (middle-left), Button Window (bottom), Colormap Window (center), Sensor Window (top-right), Image Window (center-right).

4 Acknowledgements

Visual's development effort has spanned several years and has been supported through multiple agencies. Contributing contracts include NRL N00014-92-C-6028, ONR N00014-95-11237, NSF OCE-9314697, NELO N141756-95-C-4806, UK/NSF OCE-9627160.

5 References

[1] Ballard, R., Jason Project, Guaymas 1993

[2] Humphris, S.E. and Kleinrock, M.C., 1996. Detailed morphology of the TAG active hydrothermal mound: Insights into its formation and growth. Geophys. Res. Lett., 23. 3443-3446

[3] Humphris, S.E., Fornari, D.J., Parson, L.M., German, C.R. and the LUSTRE '96 Team, 1996. Geologic and tectonic setting of hydrothermal activity on the summit of Lucky Strike seamount (37 17' N, Mid-Atlantic Ridge). EOS, Trans. Amer. Geophys. Un., 77, 699.

[4] Fornari, D.J., Humphris, S.E., Parson, L.M., Blondel, P., German, C.R. and the LUSTRE '96 Team, 1996. Detailed structure of Lucky Strike seamount based on DSL-120 kHz sonar, ARGO-II and ROV Jason Studies. EOS, Trans. Amer. Geophys. Un., 77, 699.

[5] Lerner, S., Data Monitoring, Access, and Analysis Systems for the MV Derbyshire Survey, 1997, Proceedings Volume 2 MTS/Ocean Community Conference '98, pp. 1109-1113, Baltimore, MD, Nov '98.

[6] Bowen, A., et al, The Woods Hole Oceanographic Institution's Remotely-Operated and Towed Vehicle Facilities for Deep Ocean Research, WHOI Technical Report, July 22, 1993.

[7] Lerner, S., Yoerger, D., Crook, T., Navigation for the Derbyshire Phase2 Survey, Woods Hole Oceanographic Institution, Technical Report, WHOI-99-11, 1999.

DOCUMENT LIBRARY

Distribution List for Technical Report Exchange – July 1998

University of California, San Diego
SIO Library 0175C
9500 Gilman Drive
La Jolla, CA 92093-0175

Hancock Library of Biology & Oceanography
Alan Hancock Laboratory
University of Southern California
University Park
Los Angeles, CA 90089-0371

Gifts & Exchanges
Library
Bedford Institute of Oceanography
P.O. Box 1006
Dartmouth, NS, B2Y 4A2, CANADA

NOAA/EDIS Miami Library Center
4301 Rickenbacker Causeway
Miami, FL 33149

Research Library
U.S. Army Corps of Engineers
Waterways Experiment Station
3909 Halls Ferry Road
Vicksburg, MS 39180-6199

Marine Resources Information Center
Building E38-320
MIT
Cambridge, MA 02139

Library
Lamont-Doherty Geological Observatory
Columbia University
Palisades, NY 10964

Library
Serials Department
Oregon State University
Corvallis, OR 97331

Pell Marine Science Library
University of Rhode Island
Narragansett Bay Campus
Narragansett, RI 02882

Working Collection
Texas A&M University
Dept. of Oceanography
College Station, TX 77843

Fisheries-Oceanography Library
151 Oceanography Teaching Bldg.
University of Washington
Seattle, WA 98195

Library
R.S.M.A.S.
University of Miami
4600 Rickenbacker Causeway
Miami, FL 33149

Maury Oceanographic Library
Naval Oceanographic Office
Building 1003 South
1002 Balch Blvd.
Stennis Space Center, MS, 39522-5001

Library
Institute of Ocean Sciences
P.O. Box 6000
Sidney, B.C. V8L 4B2
CANADA

National Oceanographic Library
Southampton Oceanography Centre
European Way
Southampton SO14 3ZH
UK

The Librarian
CSIRO Marine Laboratories
G.P.O. Box 1538
Hobart, Tasmania
AUSTRALIA 7001

Library
Proudman Oceanographic Laboratory
Bidston Observatory
Birkenhead
Merseyside L43 7 RA
UNITED KINGDOM

IFREMER
Centre de Brest
Service Documentation - Publications
BP 70 29280 PLOUZANE
FRANCE

50272-101

REPORT DOCUMENTATION PAGE	1. REPORT NO. WHOI-99-13	2.	3. Recipient's Accession No.

4. Title and Subtitle Visual: A Visualization System for Accessing and Analyzing Multi-Sensor Data	5. Report Date August 1999
	6.
7. Author(s) Steve Lerner	8. Performing Organization Rept. No. WHOI-99-13
9. Performing Organization Name and Address Woods Hole Oceanographic Institution Woods Hole, Massachusetts 02543	10. Project/Task/Work Unit No.
	11. Contract(C) or Grant(G) No. (C) OCE-9627160 (G)
12. Sponsoring Organization Name and Address National Science Foundation	13. Type of Report & Period Covered Technical Report
	14.

15. Supplementary Notes

This report should be cited as: Woods Hole Oceanog. Inst. Tech. Rept., WHOI-99-13.

16. Abstract (Limit: 200 words)

Visual is a visualization system used to access and analyze high-volume multi-sensor data collected from remotely operated underwater vehicles. Since 1991, scientists have used Visual for scientific visualization and analysis of underwater surveys ranging from real-time survey monitoring, to geological mapping and interpretation of hydrothermal vent sites, to a forensic study of a shipwreck. This report describes Visual's capabilities and gives examples of typical applications for Visual including sonar visualization, real-time monitoring, and multi-sensor data access and analysis. This report also includes a User's Manual and Reference Guide for the Visual system.

17. Document Analysis a. Descriptors

Multi-Sensor Data Analysis and Visualization
Real-Time Survey Monitoring
Sonar Visualization

b. Identifiers/Open-Ended Terms

c. COSATI Field/Group

18. Availability Statement Approved for public release; distribution unlimited.	19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages 50
	20. Security Class (This Page)	22. Price